Doctoral Thesis

Learning to recommend
Interactive learning with limited feedback

Author(s):
Vanchinathan, Hastagiri P.

Publication Date:
2015

Permanent Link:
https://doi.org/10.3929/ethz-a-010717341

Rights / License:
In Copyright - Non-Commercial Use Permitted
LEARNING TO RECOMMEND: INTERACTIVE LEARNING WITH LIMITED FEEDBACK

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES OF ETH ZURICH
(DR. SC. ETH ZURICH)

presented by
HASTAGIRI PRAKASH VANCHINATHAN
Master of Science in Computer Science
(Indian Institute of Science, Bangalore)

born on 30 June 1983

citizen of India

accepted on the recommendation of
Prof. Dr. Andreas Krause, examiner
Prof. Dr. Thomas Hofmann, co-examiner
Prof. Dr. Yisong Yue, co-examiner

2015
ABSTRACT

In many problems in online machine learning, we receive only partial feedback in each round. This setting is particularly important to model many real world problems including recommender systems, experimental design and dynamic pricing. Especially in cases where the set of actions available is large, there is also a need to model similarity in order to generalize well from sparse feedback. Existing approaches to many of these problems usually have little empirical evaluation or apply heuristics with no performance guarantees.

In this thesis we develop principled techniques and algorithms to tackle the problem of online learning with partial feedback. We use ideas from Gaussian process optimization and Bayesian inference to design statistically efficient feedback sharing mechanisms. We propose computational and statistical techniques to speed up our algorithms and show that they scale easily to web scale data.

We consider the problem of personalized online recommender systems and posit it as a contextual multi-arm bandit list selection problem. For this setting, we develop a general algorithm, CGPRank that effectively shares feedback across (1) list positions, (2) items, and (3) users.

We then consider the problem of interactive set selection to maximize cumulative value. In this setting, we formalize a novel class of problems called AVID - Adaptive Valuable Item Discovery. We consider settings where there is a need to diversify the set of selected items in the AVID setting. We also consider a generalization of AVID where there are item specific costs of selection. For these settings, we develop a family of algorithms, GP-Select that can generalize feedback efficiently using Gaussian processes and diversify using submodular maximization techniques.

Finally, we introduce a new family of algorithms, BPM, for locally observable stochastic partial-monitoring problems. We incorporate prior information about the outcome distribution in the form of a confidence ellipsoid that allows us to effective share feedback among actions. We present two versions of BPM– BPM-TS and BPM-Least that differ in the action selection process in each iteration.
For each of the algorithms that we present, we prove strong performance guarantees in terms of the regret, which measures the difference in utility with respect to an omniscient algorithm.

In addition to algorithms and theoretical guarantees, we also evaluate our algorithms on large scale real world data. Specifically,

1. We evaluate CGPRank on two real world recommender systems tasks with datasets from Yahoo! and Google books improving performance by 18 % in one setting.

2. We evaluate GP-Select on three real world datasets from (1) flight pricing industry, (2) drug design, and (3) recommender systems.

3. We evaluate BPM on a dynamic pricing dataset with data collected from Amazon Mechanical Turk platform.

In dieser Arbeit entwickeln wir Techniken und Algorithmen um das Problem des online Lernens mit partiellem Feedback anzugehen. Wir benutzen Ideen der Optimierung mit Gauss’schen Prozessen und Bayes’scher Inferenz, um statistisch effiziente „feedback sharing“ Mechanismen zu entwickeln. Wir schlagen Techniken vor um unsere Algorithmen zu beschleunigen und zeigen, dass sie mit Daten der Größenordnung des Webs umgehen können.

Wir betrachten das Problem der personalisierten Empfehlungsdienste und formalisieren es als ein „contextual multi-arm bandit list selection“ Problem. Für dieses Szenario entwickeln wir einen Algorithmus, CGPrank, welcher effektiv Feedback über (1) Listenpositionen, (2) Elemente, und (3) Benutzer hinweg teilt.

Wir betrachten weiterhin ein neuartiges Problem der interaktiven Mengenselektion, genannt AVID – Adaptive Valuable Item Discovery. Wir betrachten auch eine Generalisierung von AVID, wo es elementspezifische Kosten für die Selektion gibt, sowie wo auf Diversität Wert gelegt wird. Für dieses Szenario entwickeln wir eine Familie von Algorithmen, GP-Select, welche Feedback durch Gauss-Prozesse effizient generalisieren und mittels submodularer Maximierungstechniken diversifizieren können.

Schliesslich führen wir eine neue Familie von Algorithmen ein: BPM für lokal beobachtbare stochastische „partial-monitoring“ Probleme. Wir zeigen wie a priori Informationen über die Ergebnisverteilung in der Form eines Konfidenz-Ellipsoids genutzt werden können, um ef-

The following publications were produced from the contents discussed in this thesis. The exposition and text in parts of this thesis are based on these:


Furthermore, the following publication was part of my PhD research but is not covered in this thesis. The topic of this publication is outside of the scope of the material covered here:

ACKNOWLEDGMENTS

I would like to thank everyone who helped me in various ways during the course of this thesis. My advisor – Andreas Krause is a fantastic role model for the budding researcher. I was fortunate to be able to learn from him from such close quarters. His quality of work, clarity of thought, level of detail, and breadth of knowledge left me incredulous and grateful throughout. He not only helped me in the formulation and orientation of my doctoral research but also instilled in me the desire to be a good teacher. Andreas was the perfect advisor one could hope for – professionally and personally. I am indebted to him not only for the research reasons but also for giving me all the space and help I needed during some tough personal challenges.

I would like to thank the members of my committee – Thomas Hofmann and Yisong Yue for valuable comments and useful feedback on the material in this thesis. I will continue to benefit from Thomas’s insights in my next stint as well. Joachim Buhmann was a teacher, mentor and constant supporter. I enjoyed not just the friendly banter but also many research and philosophical discussions with him. I want to thank my Masters thesis advisor – Narahari Yadati for all the valuable guidance and constant encouragement to pursue a PhD degree.

I would like to thank my mentors - Deepak Agarwal, Souvik Ghosh and Noam Koenigstein, other collaborators and colleagues during my internships at LinkedIn and Microsoft R&D. I also had the opportunity to mentor and learn from an amazing set of Masters thesis students – Isidor, Ilija and Nedyalko. I would also like to thank my other external collaborators – Oren Sar Shalom, Ulrich Pacquet, Andreas Marfurt, Donald Kossmann, Fabio De Bona and Charles-Antoine Robelin.

Rita was a boon to have in the group as she smilingly took on every administrative task we piled on her. Her ever welcoming attitude and warm personality made life so much easier for the students and researchers. I also want to thank all the current and former members of Learning and Adaptive Systems group including Yuxin, Adish, Gábor, Baharan, Mario, Amin, Alkis, Josip, Olivier, Hamed, Sebastian, Felix and Jens. I would remember fondly the many coffee breaks, discussions,
travels, shared teaching responsibilities, reading groups, deadline day pizzas, ...

I believed it would be difficult at my age to make close friends out of initially professional acquaintances. I was proven wrong over and over again during the course of this thesis. Sharon helped me out in so many ways and acted as a sounding board for both research and personal stuff. David was a fantastic person to be around with. I learnt very many things from him and I am glad to have collaborated with him. Most importantly, Sharon and David made life in Zurich so much better with all the hanging out, partying, drinking, gossiping and all other assorted nonsense. Gábor was a good collaborator and a better friend. Thanks for the squash, coffee breaks and for very many things. Patrick encouraged me to apply to ETHZ and was instrumental in my decision to move to Switzerland. He was helpful in many ways throughout my time in Zurich and remains a good friend. Yuxin was a fantastic office mate and a great friend. I enjoyed his company during the daily work, many discussions, shared deadlines and conference trips. I am grateful for all the fun and general camaraderie I shared with Adish, Gabriel (special thanks for the German translation of the thesis abstract), Martin, Xeno, Mario, Brian, Ilija, Olivier, Kate and Cheng.

Apart from professional connections, so many other friends in Zurich helped me out and/or made life more fun. In particular, Aparna and Ankit – thanks for being almost family here in Zurich. Kriti and Tejaswi – so glad you guys moved here. John – for always being around and for being the lame president of the cricket club. Mukta – for the lunches and the arguments. The entire Winterthur Cricket Club family – for amazing games, social evenings and fantastic tours.

Friends around the world supported and encouraged me during the thesis. Ganesh, Anand and Arvind joined in for trips, conversations (in person and via Skype/IM). Niranjan and Divya hosted me for a while and were fun company during my internship. Deepak, Radhani, Annie, Raghavan, Janani all made my visits to the US enjoyable and memorable during my internship and conferences. Vibhu was a constant source of encouragement and support. Ashok was an inspiring role model and pointed out ETHZ as a good venue for my PhD. Rajesh remained a good friend and supporter. Balaji and Shradha hosted me multiple times during my visits to Bangalore. Mayaram, Siddhu, Sushil, Archana and Anna chimed in during good and bad times and for moral support.
My cousins were a source of great support and fun throughout. I enjoyed my time during travels, dinners and phone/skype calls with them. Balaji and Shwetha, Dhyanesh and Janani, Shyam and Hamsa (and the kids), Arun and Irani were all either hosts or guests (or both) during the course of this thesis.

I will also be grateful to Arsene Wenger and Arsenal FC for the engrossing football and also for falling out of the title race early enough in the Premier league and the Champions league so that I could concentrate on the annual fixture of the NIPS deadline.

A big thanks to my beloved parents for raising me with the spirit of inquiry and curiosity and letting me follow my dreams throughout. A world of thanks to my sister, Preethi. She was the rock on which I could lean on anytime and was my constant source of support, encouragement and motivation. My life was made infinitely better with the arrival of my wife and friend, Pavithra. I thank her for her love, patience, understanding and tolerance during the course of this thesis (and beyond!). Huge hugs and kisses to my dearest Haasini whose impending arrival served as valuable motivation for me to complete my thesis.
# CONTENTS

1 INTRODUCTION ............................................. 1
   1.1 Thesis Goal and Contributions .................. 3
   1.2 Outline of the Thesis ................................. 6

2 BACKGROUND .................................................. 7
   2.1 Online Learning ......................................... 7
   2.2 Multi-Arm Bandit ....................................... 9
      2.2.1 Setup ............................................. 10
      2.2.2 Early Approaches ................................. 11
      2.2.3 Upper Confidence Bounds ....................... 12
      2.2.4 Randomized Probability Matching ............... 13
      2.2.5 Large Action Spaces ............................. 14
      2.2.6 Contextual Bandits .............................. 15
      2.2.7 Other MAB settings .............................. 18
   2.3 GP–UCB family ........................................... 19
      2.3.1 Bayesian Optimization ............................ 20
      2.3.2 Gaussian Processes ............................... 20
      2.3.3 GP-UCB ........................................... 21
      2.3.4 CGP-UCB .......................................... 23
      2.3.5 GP-BUCB ........................................... 24
   2.4 Other settings with limited or expensive feedback .. 24
      2.4.1 Partial Monitoring ................................ 24
      2.4.2 Label Efficient Prediction ....................... 25
   2.5 Summary .................................................. 26

3 STATISTICALLY EFFICIENT WEB SCALE RECOMMENDATIONS 27
   3.1 Introduction ............................................. 27
      3.1.1 Related Work ...................................... 28
   3.2 Insights from a Large Scale Recommender System .... 30
      3.2.1 Inter-item Similarity Interactions .............. 30
      3.2.2 Item Fatigue ....................................... 32
      3.2.3 CTR Variations by Time of Day ................. 33
   3.3 Problem Setup ............................................ 34
   3.4 Algorithm and Analysis ................................. 36
      3.4.1 Sharing feedback across positions ............... 36
      3.4.2 Sharing across items/contexts via kernels .... 37
      3.4.3 Explore-Exploit in List Selection ............... 39
## CONTENTS

3.4.4  Computing the Tradeoff Parameter .................................. 41  
3.4.5  Regret Analysis of CGPRANK ................................. 42  
3.5  Scaling to Web Scale Recommendation Tasks ..................... 44  
3.6  Experimental Setup .................................................. 46  
3.6.1  Yahoo! news article recommendation ....................... 47  
3.6.2  Google e-books recommendation ............................. 51  
3.7  Results ............................................................... 52  
3.8  Summary ............................................................... 54  

4  ADAPTIVE VALUABLE ITEM DISCOVERY .... 55  
4.1  Introduction .......................................................... 55  
4.2  Problem Setup ....................................................... 58  
4.2.1  Regularity Assumptions .................................. 58  
4.2.2  Explore-Exploit Tradeoff ................................. 59  
4.3  The GP-Select algorithm ........................................... 60  
4.3.1  Regret bounds .................................................. 61  
4.3.2  Interpretation of the Theorem .......................... 64  
4.3.3  Similarities between CGPRANK and GP-Select .... 65  
4.4  Experiments .......................................................... 66  
4.4.1  Case Study I: Airline Price Update Prediction Task .... 66  
4.4.2  Case Study II: Vaccine Design Task ..................... 67  
4.5  Summary ............................................................... 69  

5  ADAPTIVE VALUABLE ITEM DISCOVERY - EXTENSIONS .... 71  
5.1  Scaling to Web Scale Data ......................................... 71  
5.1.1  Experimental Setup ....................................... 72  
5.1.2  Results ......................................................... 74  
5.2  Selecting Diverse Subsets ......................................... 74  
5.2.1  Submodular Functions and Diversity ................ 74  
5.2.2  Algorithm and Analysis ................................. 75  
5.2.3  Experimental Results ................................ 79  
5.3  Non-Uniform Costs .................................................. 81  
5.3.1  Algorithm and Analysis ................................ 81  
5.3.2  Experimental Results ................................ 84  
5.4  Summary ............................................................... 85  

6  BAYESIAN PARTIAL MONITORING .................. 87  
6.1  Introduction .......................................................... 87  
6.2  Problem Setup ....................................................... 89  
6.3  BPM family of Algorithms ...................................... 92  
6.3.1  Update Rule ..................................................... 93  
6.3.2  Selection rules ................................................. 94  

xiv
CONTENTS

6.4 Analysis ............................................. 95
6.5 Experiments ........................................... 100
  6.5.1 Implementation Details ......................... 100
  6.5.2 Simulated dynamic pricing games .......... 101
  6.5.3 Results on real data ......................... 103
6.6 Summary ............................................. 104

7 DISCUSSION AND CONCLUSION ........................ 105
  7.1 Summary ......................................... 106
    7.1.1 Statistically efficient web scale recommendations 106
    7.1.2 AVID — AdaptiveValuableItemDiscovery ...... 107
    7.1.3 Bayesian partial monitoring ................. 107
  7.2 Future Directions ................................ 108
    7.2.1 Contextual recommendations ............... 108
    7.2.2 Adaptive Valuable Item Discovey .......... 108
    7.2.3 BPM ........................................ 109

BIBLIOGRAPHY .......................................... 111
INDEX .................................................. 125
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Multi–Stage Drug Design</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Xbox 360 Main dash recommendations</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Yahoo! Today module and Google Books – Recommendation Datasets used</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>XBox 360 dataset – click probability (actual and estimated) against predicted rating and similarity for batch size 2</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>XBox 360 dataset – Impressions before a click</td>
<td>32</td>
</tr>
<tr>
<td>3.3</td>
<td>a: Performance of CGPRank for single item selection on Yahoo! Today Module, b: Performance of CGPRank for the Google Books dataset</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>Contextual list selection task with $b = 4$</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>a: Position effects for $b=2$, b: Average regret per slot</td>
<td>48</td>
</tr>
<tr>
<td>4.1</td>
<td>a: Average Regret in vaccine design task, b: Flight ticket price change prediction</td>
<td>65</td>
</tr>
<tr>
<td>5.1</td>
<td>Experimental evaluation (a) and algorithm speedup (b) on the news recommendation dataset</td>
<td>73</td>
</tr>
<tr>
<td>5.2</td>
<td><em>Balancing utility and diversity</em>: Illustration of sets selected for trading $f$ against $D$ when varying parameter $\lambda$</td>
<td>77</td>
</tr>
<tr>
<td>5.3</td>
<td>a: Average Regret (Diversity), b: Effect of Inducing Diversity</td>
<td>80</td>
</tr>
<tr>
<td>5.4</td>
<td><em>Non-uniform costs</em>: Comparison of GP-Select with the baselines for the vaccine design task under non-uniform item costs.</td>
<td>85</td>
</tr>
<tr>
<td>6.1</td>
<td>a: Cell decomposition, b: Before the update, c: After the update</td>
<td>92</td>
</tr>
</tbody>
</table>
List of Tables

Figure 6.2  
a, b, d, e: Comparing BPM on the locally non-observable game,  
c: Effect of a misspecified prior,  
f: Performance of the BPM on the real dynamic pricing dataset  

LIST OF TABLES

Table 3.1  
Percentage of users with significantly higher CTR (compared to average CTR across whole day) in various time slots. Note that a subset of users could feature under multiple time slots where others might not feature in any timeslot.
COLLABORATIONS

With the fast pace of research in Machine learning and the breadth of the area, it is natural that one tends to collaborate with many researchers during the graduate student career. I am no exception and was fortunate to have a great set of collaborators. I mention here the collaborators grouping them by parts of the thesis where their contribution was invaluable. Needless to mention, my advisor, Prof. Andreas Krause was involved in all the research presented in this thesis, providing valuable suggestions, ideas and technical mentoring.

1. **Chapter 3**: Isidor Nikolic conducted the experiments on the Google books dataset as part of his Master’s thesis while he was an intern at Google, Zurich. Fabio de Bona was his mentor at Google who provided valuable input to carry out the experiments and participated in discussions about the work. Isidor Nikolic was supervised in his thesis by Prof. Andreas Krause and me. The insights from the Xbox recommender system were obtained with the collaboration of Noam Koenigstein and Oren Sar Shalom during a research visit to Microsoft, Israel.

2. **Chapter 4**: Andreas Marfurt carried out the experiments on the Amadeus airline pricing dataset as part of his Master’s thesis. Charles-Antoine Robelin of Amadeus provided valuable input on the dataset and motivation for the project. Andreas Marfurt was supervised in his thesis by Prof. Andreas Krause and Prof. Donald Kossmann.

3. **Chapter 6**: The work in this chapter was carried out in close collaboration with Gábor Bartók who also played a major role in the analysis of the algorithms presented. Adish Singla also provided the Amazon mechanical turk dataset on which the experiments were conducted.
INTRODUCTION

The field of machine learning is focussed on designing efficient algorithms that can model, learn from and predict data arising from a multitude of real world problems. An important area of machine learning attempts to learn from and predict sequentially arriving data points where the predictions can only be partially verified. This thesis is focussed on such problems collectively known as Online learning with partial feedback. For these tasks, our goal is to develop statistically and computationally efficient learning algorithms with provable performance guarantees.

Consider the following examples:

1. **Recommendation Systems**: Recommender systems are ubiquitous in today’s online shopping world. Popular examples include online stores like Amazon and eBay, online entertainment services like Netflix and Spotify, social media websites like Facebook, LinkedIn and Twitter and digital goods stores like Xbox store, Apple iTunes and Google Play. The common goal in these examples is to increase user satisfaction by recommending items (products, movies, people, etc.) that align with the interests of the user. However, in order to learn the interests of the user well, the system has to recommend many different items, some of which the user might not like. This leads to a conundrum whereby the designer of the service has to choose between recommending items of potentially high value and recommending items that give more insight into the user preferences.

2. **Dynamic Pricing**: Dynamic pricing is a process of revenue maximization pursued by the seller of a good or service wherein a price for the item is quoted based on the perceived value it adds to the buyer. It is a widely used practice in the online travel and hospitality industry. It is also used when there are multiple copies of items to be sold to a large consumer base. An example would be a digital goods store like Apple iTunes or Google Play. Here the seller faces a choice between setting a “safe” but low price at
Introduction

Target Identification
Target Validation
Further Tests

Figure 1.1: Multi-Stage Drug Design: The challenge is to reliably and cheaply identify good candidates in the early stages before settling on the best choice through further testing. The target validation phase can be modeled as a problem of online learning with partial feedback

which she is confident of completing the sale and setting a risky but high price at which the sale might not happen but if it does, it realizes more profit for her.

3. Drug Design: Drug design is the process of developing a medication or vaccine using knowledge of the target site in the body that the drug is supposed to affect. Typically the drug designer has a large number candidates to choose from initially. The actual choices are based on a series of potentially expensive tests over multiple stages. The time to synthesize a useful drug and its final cost thus depends on the number and costs of tests in the design phase. Here, the designer faces a problem of advancing in the pipeline with a few promising candidates or testing more candidates in the hope of finding the best possible drug.

In all the above examples, the common dilemma is deciding between acting on the best current knowledge or learning more by experimenting. This dilemma is widely known as the exploration – exploitation tradeoff. Learning in this area is typically online in nature (that is, the learning happens in rounds or epochs). In each round, the learner has to choose between one of many actions and receives feedback only for a small subset of actions. Performing well in this setting means quickly learning the optimal action(s) in the presence of this partial feedback. The problem becomes statistically more difficult when the number of choices available in each round is large. In this case, we
need to model the similarity between actions and share the feedback among similar actions. All the problems discussed in this thesis involve tackling this dilemma in various forms and solving it in a principled manner. Since our research is tailored towards problems of large data, computational feasibility and efficiency are important concerns in the solutions provided.

1.1 THESIS GOAL AND CONTRIBUTIONS

We seek to design theoretically sound and computationally efficient learning algorithms to learn under the presence of partial feedback. One of the common themes of our work in this thesis is the ability to generalize from limited feedback events to a large number of items or users. We achieve this by modeling the feedback function as “smoothly” varying over the items. This allows us to share feedback among similar items. Apart from statistical efficiency and provable guarantees, we also seek to provide computationally efficient solutions. To this end, we test our algorithms on many real world settings with large data. We summarize our contributions in the following:

1. First, we present insights into a large scale recommender system by discussing a few examples of the challenges faced in the front dash module of the Microsoft Xbox recommender system (See Figure 1.2). We then consider the formal problem of contextual recommendations in web portals and stores. We have an inventory of items, \( V = \{v_1, \ldots, v_n\} \). At each round \( t \) we are given a context \( z_t \in Z \) (as examples, \( z_t \) could be the user profile features or a
node in a social graph $Z$) and have to chose a list $L_i$ of $b$ items. Since we are dealing with recommendation problems on the web, the ordering of the list is also important as the order of items influences the interactions the user has. A good recommendation list for a given user/context also has to take recent feedback into account. The main challenges in this setting are threefold. (1) There is an exponential number of lists to choose from; (2) the number of items available for recommendation is often large, compared to the relatively sparse click feedback; (3) optimality of the ordering depends on the context which in itself could be from a large set. In order to tackle this problem, we present CGPRANK. Using ideas from Gaussian Process optimization and prior work in learning with limited feedback, CGPRANK efficiently tackles the exploration – exploitation tradeoff. We prove strong performance guarantees for CGPRANK. We evaluate our approaches on two large datasets from real world recommendation task datasets provided by Google ebooks store and Yahoo! (See Figure 1.3).

2. We present a novel class of online learning problems - AVID - Adaptive Valuable Item Discovery that capture many practical problems faced in budgeted experimental design under partial feeback. In its simplest version the problem is defined by a set of items $V = \{v_1, \ldots, v_n\}$, a cardinality constraint $b$ which captures the total number of items that can be selected, an unknown function $f$ that assigns values to the items and a sequence of feedback events $y_t = f(v_i) + \epsilon_t$, which are the noisy observations of value of items $v_i$ selected in round $t$. The goal is to maximize

---

**Figure 1.3:** Datasets used for the recommendation tasks in Chapter 3: On the left, news article recommendation on the Yahoo! Today module. On the right, book recommendations on the Google books store.
cumulative value of the items selected while selecting less than \( b \) items. This abstract problem finds application in experimental design, drug design, recommendation systems, etc. Another problem in the AVID family is value maximization where each item has an individual cost of selection and there is an overall budget constraint \( b \) on the cumulative cost of the items selected. This problem generalizes the popular Knapsack problem to the setting where the values of the items are not known a priori. We also discuss an useful extension to both these problems where we want to induce diversity in the selected subset. For this class of problems, we provide a learning algorithm–GP-Select with strong theoretical guarantees. We test the efficacy of GP-Select on an online flight price update dataset and a drug design dataset. We discuss techniques for scaling of GP-Select to large web-scale datasets and demonstrate the improvements on Yahoo!’s news recommendation dataset.

3. We consider stochastic Partial Monitoring, a repeated game where in every time step a learner chooses an action while, simultaneously, an opponent chooses an outcome. Then the player is credited a loss based on the action and outcome chosen. The learner also receives some feedback (a symbol from an abstract alphabet) based on which she can make better decisions in subsequent time steps. The player does not observe the loss but can only see the feedback symbol. The goal of the learner is to minimize her cumulative loss (unobserved) over some time horizon. Of course, if there is no correlation between the loss and the feedback there is no hope to minimize the loss. A popular and useful example of this problem class dynamic pricing as discussed above. We propose a new family of algorithms - BPM that uses a Bayesian update rules to effectively track the losses and find the best action. Algorithms in this family are able to generalize the restricted feedback obtained across multiple actions due to modeling of similarity between the actions. We present two versions of BPM: BPM-Least and BPM-TS that use slightly different intuitions to track the losses. Apart from simulated experiments, we also test the efficiency of BPM on a dynamic pricing dataset collected through crowdsourced data.
1.2 OUTLINE OF THE THESIS

The rest of the thesis is organized as follows: In Chapter 2, we present an overview of existing literature in the field of online learning under partial feedback. We also provide a brief review of some of the concepts we use throughout the rest of the thesis. The notation introduced in Chapter 2 is also consistent throughout the thesis and any new notation or deviations are explained when used.

In Chapter 3, we present insights gleaned from the Xbox recommender system. We formulate the online recommendations problem and present details of CGPRank along with theoretical guarantees and experiments.

In Chapter 4, we present the basic formulation of AVID - Adaptive Valuable Item Discovery. We present our algorithm, GP-Select that solves the AVID problem in the uniform cost case with cardinality constraint. We offer theoretical insights and empirical evidence for GP-Select’s efficacy.

In Chapter 5, we first show we can scale GP-Select to very large datasets. We provide insights for computational improvements and demonstrate efficacy on a large scale dataset. We then formulate a version of AVID with diversity inducing selection function that prefers valuable and diverse sets. Finally, we present the most general setting of AVID where each item can have different costs of selection. We show that GP-Select can be extended to these settings with strong performance guarantees and also demonstrate the efficacy of GP-Select via experiments.

In Chapter 6, we present the BPM family of algorithms for stochastic partial monitoring problems. We present two algorithms - BPM-Least and BPM-TS and show the effectiveness using theoretical guarantees and experiments.

Finally, in Chapter 7, we provide a concluding discussion of the work in this thesis and provide pointers to what we believe are useful and interesting future directions.
BACKGROUND

In this chapter, we provide the methodological background required to understand the work in the later chapters. We also provide a high level introduction to some of the modeling tools we use throughout this thesis. We begin with the field of online learning and delve deeper into the problem of limited feedback in online learning which include Multi–Arm Bandits and its generalizations including Partial Monitoring. In each of these problems, we introduce the notion of regret which is the performance measure we use repeatedly in this thesis. In most cases, we refer only to few key pointers to literature that act as a sample for the topics in the discussion. This is by no means a complete review of the work in these areas which would be too vast to cover in this chapter.

2.1 ONLINE LEARNING

Traditionally, supervised machine learning techniques work under the assumption of access to a training dataset \( \mathcal{D} \), which consists of labeled instances of the form \((x_i, y_i)\), where \(x_i\) is a sample from the domain \(\mathcal{X}\) (for example, \(\mathcal{X} \subseteq \mathbb{R}^d\)) and \(y_i \in \mathcal{Y}\) is its label as assigned by an expert or supervisor. Typically, \(y_i\) is categorical if it is a classification task or real valued if it is a regression task.

However, in many real world problems, we do not have access to these training instances all at once but the data arrives in a stream. Online (or sequential) learning is the branch of machine learning that deals with such problems (See Cesa–Bianchi and Lugosi (2006) for a detailed introduction and discussion and Shalev–Shwartz (2011) for connections to convex optimization). Typically, online learning proceeds in rounds where in each round \(t\) we receive a instance, \(x_i\) and have to predict a label or value in response. We have access to a set of blackbox predictors \(V = \{v_1, \ldots, v_n\}\) where \(v_i : \mathcal{X} \rightarrow \mathcal{Y} \forall i \in \{1, \ldots, n\}\). In each round we pick \(v_{i_t} \in V\) and predict \(\hat{y}_t = v_{i_t}(x_t)\). We then receive the true label or value of \(x_t\) as \(y_t\) and suffer a loss \(\ell(y_t, \hat{y}_t)\). The objective then is to minimize the cumulative loss over time \(\mathcal{L}_T = \sum_{t=1}^{T} \ell(y_t, \hat{y}_t)\). \(v^* \in V\) is the fixed predictor that achieves the least cumulative loss.
Note that $v^*$ is only identifiable in hindsight after the entire stream of data has been observed.

A stylized procedure of online learning:

- For $t = 1, 2, \ldots, T$
- Receive $x_t \in \mathcal{X}$
- Choose $v_{i_t} \in V$
- Predict $\hat{y}_t = v_{i_t}(x_t)$
- Receive true label $y_t$
- Suffer loss $\ell(y_t, \hat{y}_t)$

**Regret**

Equivalent to minimizing the cumulative loss, it is instructive to consider the regret of a learning algorithm. The regret measures the additional cumulative loss suffered by the algorithm over the loss suffered by the best fixed predictor, $v^* \in V$. Cumulative regret (or simply, regret) is the preferred metric for reporting performance of online learning algorithms and we will be using this in all further chapters to characterize the performance of our algorithms. If $R_T$ is the $T$ round regret of a learning algorithm,

$$R_T = \sum_{t=1}^{T} \ell(y_t, \hat{y}_t) - \sum_{t=1}^{T} \ell(v^*(x_t), \hat{y}_t)$$

(2.1)

In order to efficiently learn, the average per–round regret must vanish as $T \to \infty$. This desirable property of the learning algorithm is referred to as achieving Hannan consistency (Hannan 1957). An important consideration in the design of learning algorithms with low regret is the time horizon $T$. In some cases, the time horizon is infinite while in others $T$ is fixed and known in advance. Unless stated otherwise, we will talk about the infinite horizon problem and the results for the known $T$ setting can be recovered as $T \to \infty$. Algorithms whose implementation does not depend on the a priori knowledge of $T$ are called anytime learning algorithms.
Adversarial versus Stochastic feedback

There is a clear distinction in the literature based on the nature of feedback for online learning. On one hand, the feedback could be adversarial (or oblivious adversarial, to be exact) in which case, an adversary generates the labels in advance while trying to make it hard for the learning algorithm to perform well. On the other hand the feedback could be stochastic in which case, the labels are provided by a stochastic function. The reader is referred to (Littlestone et al. 1995; Vovk 1990) for seminal works in the adversarial feedback setting and also to (Cesa–Bianchi and Lugosi 2006; Rakhlin 2009) for a more comprehensive coverage of the topic. Zinkevich (2003) and LeCun and Bottou (2004) are examples of highly influential works in the stochastic feedback setting. In this thesis, we deal always with the stochastic feedback setting.

In the general framework of online learning, we note that the loss of our action or prediction is completely revealed at the end of every round. In literature, this is known as the full–information setting. However, there is important class of problems where we do not have full–information. Learning with limited feedback is the term used to refer to such settings. Online learning with limited feedback poses additional challenges of generalizing from reduced feedback. Even in this restrictive class of problems, Hannan consistency can be achieved by a variety of algorithms for different settings of this class. The rest of this chapter is devoted to an overview of this class of problems.

2.2 MULTI-ARM BANDIT

Multi–Arm bandit (MAB) problems are a class of online/sequential learning tasks where the feedback is available after each round only for the chosen action(s) and not for the other remaining choices. We restrict most of our attention to the stochastic variant of the problem. More detailed summaries of individual results and algorithms can be found in literature (Cesa–Bianchi and Lugosi 2006; Gittins et al. 2011; Tran–Thanh 2012).
2.2.1 Setup

The stochastic MAB problem was first introduced by Robbins (1952) as a setup with $k$ arms. We denote by $V$, the set of arms and by $v_t$ the arm that is chosen in round $t$. Thus for the $k$–arm bandit problem considered in Robbins (1952), we have $V = \{1, \ldots, k\}$. Each arm, when pulled, produces a reward that is independently drawn from an unknown distribution. At each step $t$, the agent (algorithm/player) can pull an arm $v_t \in V$ and receive a reward $y_t(v_t)$ (or negative loss). The agent’s objective is to maximize the cumulative rewards obtained at the end of the process of playing for $T$ rounds.

$$\max_{v_t} \sum_{t=1}^{T} y_t(v_t)$$

In the case of the stochastic MAB with finite arm set, the single best action in hindsight, $v^* \in V$ is the one which has the highest expected reward $\mu^*$. We can then consider the regret of an algorithm by comparing its cumulative reward over $T$ steps to the cumulative reward of the optimal action.

$$R_t = \sum_{i=1}^{T} y_t(v^*) - \sum_{i=1}^{T} y_t(v_t)$$

Multi–arm bandit settings were originally studied in the area of experimental design (Thompson 1933; Robbins 1952; Whittle 1980; Gittins et al. 2011) and were later used in several problems in clinical trials (Hardwick et al. 1991), computational finance (refer to Bäuerle and Rieder (2011) and references therein), recommender systems (Scott 2010) and many other areas of sequential experimentation.

The key to achieving low regret in the multi–arm bandit setting is handling the inherent tradeoff between learning more about the reward function through sampling of different arms and accumulating higher payoffs through playing arms with higher payoff potential in the current round. This dilemma is referred to in literature as exploration–exploitation tradeoff. The performance of the MAB algorithms as measured in terms of cumulative regret is considered to be acceptable if it grows sublinearly with time. This is because vanishing regret is achievable only if the cumulative regret grows slower than the number of iterations. Some MAB algorithms can be shown to achieve Hannan consistency.
(asymptotically vanishing regret) with high probability. These kind of performance guarantees, known as probably approximately correct (PAC) guarantees are common in the MAB literature. In some other cases, the regret bounds only hold in expectation. This means that $\frac{\mathbb{E}(R_t)}{T} \to 0$ as $T \to 0$.

2.2.2 Early Approaches

Some of early approaches used a simple greedy policy of picking the arm with the best estimated reward so far (see Sutton and Barto (1998) and references therein). Two variants of the greedy algorithm that have been used in a variety of applications are $\epsilon$-first (Even–Dar et al. 2002) and $\epsilon$–greedy (Watkins 1989). In $\epsilon$–first, the algorithm is randomly allowed to explore arms for $\epsilon T$ rounds and then plays the best arm so far in the remaining $T-\epsilon T$ rounds. The $\epsilon$–greedy algorithm in each round plays the best arm so far with probability $(1-\epsilon)$ and explores a random arm with probability $\epsilon$. Both these approaches suffer linear regret due to constant exploration probability. $\epsilon_n$–Greedy (Auer, Cesa–Bianchi, and Fischer 2002) which is a variant of the $\epsilon$–Greedy algorithm overcomes this problem of linear regret. In $\epsilon_n$–Greedy, the exploration probability, $\epsilon$ decreased over time as the reward expectations of the arms became more accurate. The rate of decrease is set to be $O(1/t)$ and the authors produce a logarithmic regret bound that is however, dependent on the gap between the expected reward of the best and the second best arms.

Another popular thread of research considered maintaining the current knowledge of the arms’ performance in indices. The most popular of these approaches is the Gittins’ Index (Gittins 1979; Gittins 1983; Whittle 1980; Weber et al. 1992). While many earlier approaches required the algorithm to maintain the historical rewards of all the arms in order to choose an arm to pull, Gittins’ indices allow the history of the arm to be represented as a single scalar and the decision in each round was made solely based on these values for all the arms. Gittins’ indices studies the MAB problem in a Bayesian setting assuming a prior belief about the reward structure of the arms.

While many of the above approaches worked well in practice, other approaches were proposed that had strong asymptotic convergence guarantees. For example, Lai and Robbins (1985) use a parametric
algorithm to prove logarithmic regret bounds for certain probability distributions on rewards using an algorithm called *uniformly good policy*. To do this they use the Kullback–Liebler divergence metric between the reward density $\bar{y}_j$ of any suboptimal arm $j$ and the optimal expected payoff, $\mu^*$. They then compute an *Upper Confidence Index* (UCI) for each arm. However, this approach required keeping track of the individual reward sequences of all the arms. R. Agrawal (1995a) extend the work of Lai and Robbins (1985) to prove logarithmic regret bounds for the general stochastic bandit case agnostic of the rewards generating distribution (as long as rewards were drawn i.i.d.). The major insight in this line of work is combining the observed rewards of the arms along with the uncertainty about the arms to make the pulling decision in each round. In these approaches, a key solution concept used to tackle the exploration–exploitation tradeoff is “optimism in the face of uncertainty” (proposed in Kaelbling (1993)).

### 2.2.3 Upper Confidence Bounds

In a unified analysis, Auer, Cesa–Bianchi, and Fischer (2002) study the stochastic MAB problem in a non–parametric setting and propose different variants of the existing algorithms. We already discussed the $\epsilon_n$–Greedy algorithm proposed in this paper. Another algorithm is built on the idea of UCI proposed initially in Lai and Robbins (1985) and R. Agrawal (1995a). The new algorithm is called UCB1 and is simpler to implement as it does not require the history of the rewards. However, the theoretical bounds for the regret guarantee is slightly worse than that of Lai and Robbins (1985) in terms of the leading constants. Another variant, UCB2, fixes this problem by recovering almost exactly the same bound as in Lai and Robbins (1985) but is relatively more complicated to implement than UCB1.

Specifically, UCB1 begins by picking every arm once and then uses the following selection rule in every iteration to identify the arm to pick. Here, $\bar{y}_{j,t}$ is the average reward from arm $j$, $t$ is the number of iterations so far and $n_{j,t}$ is the number of times arm $j$ has been selected until round $t$, $v_t \in V$ is the arm that is selected in the $t$-th iteration and we use the natural logarithm.
2.2 MULTI-ARM BANDIT

\[ v_t = \max_{i \in \{1, \ldots, k\}} \bar{y}_{j,t} + \sqrt{\frac{2\log t}{n_{j,t}}} \]  

(2.4)

The second term in Equation 2.4 is the size of the one–sided confidence interval that contains the true reward with high probability. An intuitive application of Chernoff–Hoeffding bounds leads to a cumulative regret guarantee of \( O(\log t) \) with high probability (hence, a PAC bound). While UCB style algorithms are much easier to implement, the actual regret bounds were worse than the UCI approach of Lai and Robbins (1985). Several other approaches were proposed in order to close this gap and prove tighter bounds on the regret (Maillard et al. 2011; Cappé et al. 2013; Kaufmann et al. 2012).

### 2.2.4 Randomized Probability Matching

Another line of attack on the exploration–exploitation tradeoff in bandit problems is that of randomized probability matching. Using this approach an arm is chosen with frequency proportional to the probability of it being optimal. One of the early variations of this scheme is the SoftMax algorithm (Luce 1959) where at each round \( t \), arm \( i \) is chosen with probability

\[ p_{i,t} = \frac{e^{\frac{y_{j,t}}{\tau}}}{\sum_{j=1}^{k} e^{\frac{y_{j,t}}{\tau}}} \]  

(2.5)

In Equation 2.5, \( \tau \) is a fixed parameter that controls the amount of exploration. As \( \tau \to 0 \), SoftMax picks greedily by reward and as \( \tau \to \infty \), it picks uniformly at random between the arms. Thus, choosing an optimal value of \( \tau \) helps to control the exploration–exploitation tradeoff. However, similar to the problem with the \( \epsilon \)-Greedy algorithm, the amount of exploration is fixed for the entire algorithm. An obvious fix to this approach is to use a time varying value of \( \tau \) that decreases as we obtain more certain estimates of the individual rewards. This is proposed as the SoftMix algorithm (Cesa–Bianchi and Fischer 1998) and is shown to achieve a \( O(\log^2(T)) \) regret which is worse than the known regret bounds for the other approaches. Another popular algorithm in this category is EXP3 introduced in Auer, Cesa–Bianchi, Freund, et al. (2002). The algorithm uses multiplicative weight updates and a
parameter $\lambda$ to trade–off between exploration and exploitation. EXP3 is designed for non–stochastic feedback and performs poorly in practice when the feedback is stochastic.

The idea of randomized probability matching in a Bayesian sense was explored quite early (Thompson 1933; Thompson 1935) and the specific algorithm became known as Thompson sampling. It found renewed interest after it was found to be very efficient in practice when applied to web scale MAB problems. Thomson sampling is an intuitive Bayesian algorithm which assumes a prior distribution on the parameters of each arm’s reward distribution. In each iteration, the arm with the highest posterior probability of being the best arm (arm with the highest reward) is selected and a new posterior is computed based on the reward. Interest in Thompson sampling was revived by the empirical successes in (Granmo 2010; Chapelle and Li 2011; May and Leslie 2011). However, there were no theoretical analysis of the algorithm until strong regret bounds were proven for the MAB problem in S. Agrawal and Goyal (2012).

Apart from the classical MAB problem setting with finite number of arms and one arm being selected in each round, there were other interesting and useful variations that were explored in literature. We discuss a few of them below.

2.2.5 Large Action Spaces

In the classical MAB setting, there were a finite number of $k$ arms, $V$ to select from in each round. Most of the regret bounds for the algorithms seen so far explicitly had the term $k$ in them. As the regret grows linearly in the number of arms, these algorithms perform poorly when the number of arms were infinite or even just finitely large. On the other hand, there were several important applications which did not fall in the finite arm setting. For example, in many online optimization problems in operations research and control, each arm could correspond to a particular set of parameter values where the values are from a finite–dimensional continuous domain. Specific applications include dynamic pricing, chemical experimental design and wireless communications.

However, it is easy to see that without further assumptions on the reward generating function it is impossible to do better as each arm will have to be pulled at least $O(1)$ times to learn about its reward distribution. One way of tackling this problem is to assume that the
reward function is globally smooth over the set of arms. In this case, experimenting with one arm provides some feedback about all the other arms. Thus, one can efficiently generalize from fewer feedback events (R. Agrawal 1995b; Kleinberg 2004; Kleinberg, Slivkins, et al. 2008; Cope 2009). A generalization of these approaches is to assume only local smoothness and generalize feedback only to subset of arms (Auer, Ortner, et al. (2007)). In these cases, smoothness (local or global) assumption means assuming that the payoff function is either locally or globally Lipschitz or Hölder continuous. In all these approaches, the set of arms was still finite but each arm corresponded to a set of parameter values that could be infinite. In Bubeck, Munos, Stoltz, and Szepesvari (2011), the authors consider a generalization of these settings where the set of arms could be any arbitrary measurable space and propose a tree–based optimization algorithm that achieves low regret. In order to do this, they assume the existence of a dissimilarity function that constrains the behaviour of the mean–payoff function. In another thread of work, it is assumed that the payoff function is linear and comes from a finite dimensional linear space. Dani, Hayes, et al. (2008) provide a near–complete characterization in terms of both upper and lower bounds. They propose two algorithms based on UCB style sampling of arms. In some special cases, they are able to obtain a regret of \( \text{polylog}(T) \) but in general they show a lower bound for \( \Theta^*(\sqrt{T}) \). Other work in literature which exploits the linear assumption include Auer (2002); Dani, Kakade, et al. (2007); Abernethy et al. (2008); Rusmevichientong and Tsitsiklis (2010); Abbasi–Yadkori et al. (2011). In general, any MAB algorithm that scales to large or even infinite/continuous action spaces has to assume existence of a similarity measure between the actions and also assume that the payoff function varies smoothly according to this similarity measure. For this setting, an important class of algorithms that assume smoothness with respect to a kernel is the GP-UCB family discussed in detail below in Section 2.3.

2.2.6 Contextual Bandits

The stochastic MAB problem is usually stateless in the sense that the algorithm faces the same decision making problem in every round. However, we could have settings where before each round, the agent receives some additional information about the context or state of the environment and then it has to select an arm from a set of arms. This
setting is referred to as multi arm bandits with side information or sometimes, contextual bandits. Consider the following example: we have a set of online news articles to recommend to a web user. Each time we recommend an item, the user could either click on the item or ignore our recommendation. Our goal is to maximize the number of clicks on the recommended articles over a stream of users who visit the website. If we did not know any additional information about the user, it is a classical MAB problem where the goal is to identify the article with the maximum click probability (referred to as clickthrough rate). However, typically websites store some information about its users (either via login information or via browser cookies). In this case, before recommending an article in each round, we also receive this additional information about the user. If we were to personalize the recommendations to the user, we could do better than naively recommending the most popular article. This problem could be modeled as a contextual bandit problem.

Typically, we receive a context \( z_t \in Z \) before every round \( t \). Note that \( Z \) could be a finite set or a metric space or any other abstract set of objects (graphs, trees etc.). If \( Z \) were a finite set of size \( |Z| \), the naive approach to a contextual bandit problem will be to run \( |Z| \) different MAB algorithm instances. However, in this case, we cannot exploit the similarity between individual contexts and the cumulative regret will always have a factor or \( |Z| \). The rewards generated in each iteration are assumed to be dependent on both the context received in that round, \( z_t \) and the arm that is selected, \( v_t \). For the stochastic setting, one can assume an unknown reward generating function \( f: Z \times V \rightarrow \mathbb{R} \) and that we observe a noisy estimate of the function’s output. Thus, the reward in each round is \( f(z_t, v_t) + \epsilon_t \)

<table>
<thead>
<tr>
<th>A stylized procedure of a contextual bandit algorithm:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• For ( t = 1, 2, \ldots, T )</td>
</tr>
<tr>
<td>• Receive context ( z_t \in Z )</td>
</tr>
<tr>
<td>• Select arm ( v_t \in V )</td>
</tr>
<tr>
<td>• Receive reward ( y_t = f(z_t, v_t) + \epsilon_t )</td>
</tr>
</tbody>
</table>

Measuring regret in the contextual bandit setting is not as simple as the non-contextual setting. This is because the actual payoffs depend
not only on the arm $v_t$ selected but also on the observed context $z_t$ in each round and a different ordering of contexts could produce a very different optimal arm sequence. Hence, instead of comparing against the best arm in hindsight, we compare the performance of the algorithm against the best policy in hindsight. For the stochastic contextual MAB, this is the maximizer over $V$ as a function of the context $z_t$. In some other settings, regret is measured against the best policy in a policy class $\mathcal{F}$. Typically, this kind of regret measure is used for the adversarial feedback setting.

Contextual bandits were first studied as a one–armed bandit with side information in Woodroofe (1979). The interest in the problem was revived from the control theory literature by work on bandits with side information (C. Wang et al. 2005). Auer, Cesa–Bianchi, Freund, et al. (2002) study the problem in an adversarial setting and propose an algorithm EXP4 with $O(\sqrt{T})$ regret that is computationally very expensive and also does not work for continuously parameterized policies. Langford and Zhang (2007) propose the epoch–greedy algorithm for the stochastic setting with much better computational efficiency but worse regret bounds ($O(T^{2/3})$). Since then, there has been an explosion of both theoretical (Beygelzimer et al. 2011; Lu et al. 2010; Slivkins 2014; Agarwal et al. 2012) and empirical studies (Li, Chu, Langford, and Schapire 2010; Li, Chu, Langford, and Xuanhui Wang 2011; Yue, Hong, et al. 2012; Tang, Rosales, et al. 2013; Tang, Jiang, et al. 2015) in the area of contextual bandits due to widespread applications in the field of online advertisements and recommender systems.

The first practical algorithm with good regret guarantees was LINUCB (Chu, Li, et al. 2011; Li, Chu, Langford, and Schapire 2010). LINUCB assumes that the features for the arms and the contexts can be together expressed in a $d$–dimensional real vector and that the reward/payoff function is linear with respect to the this feature vector. Thus, each arm $v \in \mathbb{R}^{d_1}$ and context $z \in \mathbb{R}^{d_2}$ can be combined via a vectorized tensor product to produce a combined vector $x_{v,z} \in \mathbb{R}^d$ where $d = d_1 \times d_2$. And the expected reward of the this (context, action) pair is given by

$$\mathbb{E}(y_t(z_t, v_t)) = x_{v_t, z_t} \theta^*$$

where $\theta^*$ is unknown weight vector. Thus, by this assumption, LINUCB reduces the problem of combinatorial bandit learning to one of linear
regression with partial feedback. LINUCB achieves $O(\sqrt{T})$ regret and is computationally efficient.

2.2.7 Other MAB settings

Apart from the classical setting and the few other variations discussed above, there are multiple studies on online learning problems with bandit feedback. While do not cover all the settings, it is important to briefly mention the following two settings which will be used in later chapters.

1. **Set Selection:** While in the classical MAB setting, only one arm is selected in each iteration, there have been several studies that looked at the setting where multiple arms could be selected in each round. In this setting, the number of arms to select per round $b$ is typically fixed for the entire run of the algorithm. Originally, Anantharam et al. (1987) extend the algorithm of Lai and Robbins (1985) to allow for multiple plays per round. They obtain logarithmic regret bounds that are asymptotically optimal but carry over the same disadvantages of the UCI algorithm. Cesa-Bianchi and Lugosi (2012) introduced the COMBAND algorithm in which the agent is allowed to select a combinatorial subset of the arms. Kale et al. (2010) studied the ordered and unordered subset selection problem under bandit feedback. They call it *slate bandits* and cite applications in online advertising. Specifically, they consider non-stochastic feedback (agnostic adversary) and measure regret against the best policy from a convex set of distributions. Theirs is a version of multiplicative weights algorithm using experts/policies that pick a ordered/unordered subset in each iteration. In Yue and Guestrin (2011), the authors model news recommendation as a subset selection bandit task. They assume that the topic coverage of a set of articles is a submodular function and this allows them to maximize coverage by recommending diverse articles. The feedback received in this case is a value indicating the diversity of the selected subset.

2. **Budgeted Bandits:** In some settings, there could be an arm specific cost of selection and the learning process is constrained by budget considerations. In other cases a penalty is imposed
for switching the selection from one arm to the next in subsequent iteration. For instance, R. Agrawal et al. (1988) consider the stochastic bandit problem with switching costs (fixed value $C$) and prove optimal logarithmic regret bounds for the same. Guha and Munagala (2009) extend this setting to account for switching costs that is dependent on the pair of arms that was being switched. They show that their approximation algorithm is only a factor of 3 away from the optimal solution. Madani et al. (2004) study the problem of budgeted bandits with arm specific costs and show that it is NP–Hard. Specifically, they study the problem of best arm identification under a budget constraint and provide an approximation algorithm for the same. Bubeck, Munos, and Stoltz (2009) also study the best arm identification under budget constraints. However, in their setting, the budget is not known a priori to the algorithm and hence they develop an any–time algorithm for the same problem. Gupta et al. (2011) study the MAB problem under the budgeted knapsack setting and prove strong regret bounds for the scalar budget case. Tran–Thanh (2012) study the the problem of budget–limited bandit learning and propose several algorithms based on $\epsilon$–first and UCB sampling for the setting where each arm had a different cost of pulling. The efficacy of these algorithms are also demonstrated on real world MAB tasks. In some cases, the budget and costs could be multi–dimensional. For instance, in online advertising, each ad has different costs of being served but also has a limit on the number of times it could be shown. So there is both a frequency and a budget cap for ads. This setting has been studied in the non–contextual (Badanidiyuru, Kleinberg, et al. 2013) and contextual setting (Badanidiyuru, Langford, et al. 2014) and optimal algorithms that are proposed were shown to perform well empirically as well.

2.3 GP–UCB FAMILY

Another line of work for MAB problems that is of central importance to this thesis is the GP–UCB family of algorithms. Before we discuss the algorithms in this family, it might be useful to briefly review general Bayesian optimization and introduce the concept of Gaussian processes.
Consider the problem of finding the global maximum $f(x^*)$ of an unknown function $f$. Evaluation of the function at each input $x \in X$ is expensive. The objective is:

$$x^* = \arg \max_{x \in X} f(x) \quad (2.7)$$

We could consider this problem (Kushner 1964; Jones et al. 1998) as a sequential decision making task of choosing $x_t$ where $f$ is evaluated as $f(x_t)$ at time $t$ based on past evaluations at $\{x_1, \ldots, x_{t-1}\}$.

Bayesian optimization (Mockus 1975; Mockus 2012) is a technique of solving this problem by using a Bayesian representation of the distribution over the unknown function $f$ being optimized to decide where to query next (Ghahramani 2015). A characteristic of Bayesian optimization is the representation of uncertainty about future outcome of actions and using this uncertainty to make decisions. There are obvious connections between Bayesian optimization and multi-arm bandit problems – objective in Equation 2.7 is identical to the best arm identification objective for the MAB setting. Usually, Bayesian optimization algorithms consist of two major interrelated components – (1) The surrogate function which is the posterior over the input space given the prior and the observations so far, and (2) The acquisition function which is used to decide where to sample next. All the algorithms discussed in this thesis have these two components as we take a Bayesian approach to solving sequential decision making tasks with partial feedback. For continuous spaces, Gaussian processes (Rasmussen and Williams 2005) are a popular choice to model the function $f$.

### 2.3.2 Gaussian Processes

A Gaussian process (GP henceforth) is a collection of dependent random variables, one for each $v \in V$, every subset of which is a multivariate Gaussian distributed in an overall consistent manner. Essentially, it is a distribution over functions. A $GP(\mu(v), k(v, v'))$ is completely specified by its mean function $\mu(v) = \mathbb{E}[f(v)]$ and its covariance (or kernel) function $k(v, v') = \mathbb{E}[(f(v) - \mu(v))(f(v') - \mu(v'))]$. It is common to assume that $\mu = 0$. It is mostly without loss of generality and particularly useful because it simplifies calculations. However, when we condition the GP
on data, we cannot do this and the mean \( \mu \) has to be the posterior mean function given the data. The covariance function \( k \) encodes the requisite smoothness properties of the feedback function \( f \) (assumed to be drawn from the GP) through its correlation behaviour. One of the big advantages of working with GPs is that the analytical formulae for the mean and the covariance of the posterior distribution can be computed in a simple form making it easy to implement the algorithms. For a noisy sample \( y_T = [y_1, \ldots, y_T]^T \) at points \( S_T = \{v_1, \ldots, v_T\} \), \( y_t = f(v_t) + \epsilon_t \) with \( \epsilon_t \sim N(0, \hat{\sigma}^2) \) i.i.d Gaussian noise, the posterior distribution over \( f \) is also a GP distribution with mean \( \mu_T(v) \), covariance \( k_T(v, v') \) and variance \( \sigma_T^2(v) \) given by:

\[
\begin{align*}
\mu_t(v) &= k_T(v)^T(K_T + \hat{\sigma}^2I)^{-1}y_T, \\
k_T(v, v') &= k(v, v') - k_T(v)^T(K_T + \hat{\sigma}^2I)k_T(v'), \\
\sigma_T^2(v) &= k_T(v, v),
\end{align*}
\]

where \( k_T(v) = [k(v_1, v), \ldots, k(v_T, v)] \) and \( K_T \) is the positive semi–definite kernel matrix \( [k(v, v')]_{v, v' \in S_T} \).

Equations 2.8, 2.9, 2.10 provide the analytical formulae for computing the exact posterior given observations. The interested reader is referred to Rasmussen and Williams (2005) for a detailed introduction and overview of the nature, properties and applications of GPs.

2.3.3 \textit{GP-UCB}

As we noted earlier, when the number of arms is large, countably infinite or even from a continuous set, we need to assume smoothness of the reward function over the set of arms. In many applications, linear bandits are sometimes not effective enough in capturing the non–linear nature of reward functions. On the other hand, Lipschitz continuity assumptions are too coarse grained and results in poor regret bounds. Assuming that the reward function is drawn from a Gaussian process (GP) allows fine–grained levels of smoothness in the modeling of the unknown reward function. GP assumptions lead to near–optimal regret bounds and efficient algorithms in practice.

\textbf{The Setup:} One wants to sequentially optimize an unknown function \( f : V \rightarrow \mathbb{R} \). In each round \( t \), we can pick a point \( v_t \in V \) and observe its utility, perturbed by noise, \( y_t = f(v_t) + \epsilon_t \). Because this is a bandit
problem we do not obtain or observe rewards for \( \{ v : v \in V, v \neq v_t \} \).
The goal, as usual, is to maximize cumulative reward \( \sum_{t=1}^{T} y_t \). In this case, we could again look at to minimize
the regret with respect to the optimal choice \( v^* = \arg \max_{v \in V} f(v) \). Note that this is exactly
the stochastic MAB setting. In order to model smoothness of the reward
function, it is assumed that the reward function is a sample from a
Gaussian process (discussed in Section 2.3.2).

Using the simple analytical formulae (Equations 2.8, 2.9 and 2.10)
to compute the exact posterior, Srinivas et al. (2012) develop a smart
sampling scheme to ensure that GP-UCB achieves Hannan consistency.
Essentially, GP-UCB involves two important components: Arm selection
and model update. Given feedback from the sampled locations \( \{ v_t \} \), it
is possible to update the posterior using Equations 2.8, 2.9, 2.10.
For arm selection, they propose a UCB inspired sampling scheme:

\[
v_t = \arg \max_{v \in V} \mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v)
\]

In Equation 2.11, \( \beta_t \) is a time varying parameter that controls the
tradeoff between exploration and exploitation. The actual function for
computing \( \beta_t \) depends on the exact setting that GP-UCB is applied
in. In extensions of GP-UCB, \( \beta_t \) is computed differently according to
the problem. Essentially, \( \beta_t \) must be large enough such that \( \mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v) \) contains the true function value \( f(v) \) with high probability.

The authors then prove regret bounds for three settings: when \( f \sim \text{GP}(0, k(v, v')) \) for finite action space \( D \), \( f \sim \text{GP}(0, k(v, v')) \) for general
compact \( D \), and the agnostic case of arbitrary \( f \) with bounded RKHS
norm. In fact, the GP assumption is especially useful because even
though \( f \) might not be actually drawn from a GP (it is enough that \( f 
\) has low RKHS norm), one can interpret \( f \) to be drawn from a GP and
then prove agnostic regret bounds. The RKHS \( \mathcal{H}_k(V) \) is a complete
subspace of \( \text{L}_2(V) \) of ‘smooth’ functions with an inner product \( \langle \cdot, \cdot \rangle_k \)
s.t \( \langle f, k(v, .) \rangle = f(v) \) for all \( f \in \mathcal{H}_k(V) \). By choosing appropriate kernel
functions, we can flexibly handle items of different types (vectors,
strings, graphs etc.). The ability to work with general feedback functions
with low RKHS norm broadens the applicability of GP-UCB to a much
broader and useful class of functions. In each of the cases the authors
obtain \( O(\sqrt{T}) \) regret that is crucially independent of the dimensionality
\( d \) of the input space \( V \). In order to do this, the regret bound contains a
term measuring the maximum information gain, $\gamma_T$ between the function $f$ and any subset of samples of size $T$.

$$\gamma_T := \max_{A \subset V, |A| = T} I(y_A, f)$$ (2.12)

where $I(y_A, f)$ is the mutual information between $f$ and the samples $y_A$. While exact computation of the bound is NP-hard, this can be approximately bounded using the fact that information gain is a submodular function and greedily maximizing the information gain leads to $(1 - \frac{1}{e})$ approximation of the optimal value. The authors also compute and bound the value $\gamma_T$ for common kernels such as linear kernel, Matérn kernel and Squared Exponential kernel which when used along with the regret bound, results in sublinear regret when using these kernels.

Several extensions in the GP–UCB family of algorithms are inspired by the original GP-UCB algorithm (Srinivas et al. 2012) which was for the non-contextual case. GP-UCB was extended to CGP-UCB (Krause and Ong 2011) for the contextual MAB setting and GP-BUCB (Desautels et al. 2014) for parallelizing decision making within the same round (or equivalently, handling delayed feedback). In this thesis, we propose two other extensions – CGPrank and GP-Select and show their efficacy on multiple real world datasets while also proving strong regret bounds.

2.3.4 CGP-UCB

Krause and Ong (2011) extend GP-UCB to the contextual setting. Here, the authors posit a composite kernel consisting of two kernels – $k_V$ to capture the similarity in the action space $V$ and $k_Z$ to capture the similarity in the context space $Z$. Among other possibilities the composite, $k((v, z), (v', z'))$ can be the sum of the individual kernels, $k_V(v, v') + k_Z(z, z')$ or the product of the individual kernels $k_V(v, v')k_Z(z, z')$. The authors then prove strong regret bounds and provide empirical evaluation of the algorithm. Combination kernels are fairly easy to construct and allows us to generalize over both actions and contexts. The resulting algorithm CGP-UCB carries over the analytical tractability of GP-UCB and its applicability is demonstrated on real world contextual bandit tasks.
GP-BUCB (Desautels et al. (2014) is an instance of the GP-UCB family which can be applied to MAB problems with delayed feedback or to parallelize explore-exploit tradeoffs. One interesting result in the theoretical analysis of GP-BUCB is that the cumulative regret of the batch selection algorithm increases only by a constant factor independent of the batch size, $B$. An important ingredient of the algorithm is the intra-batch variance reduction through a smart “feedback hallucination” approach that achieves better regret bounds compared to the naive approach of running $B$ separate (or chained) instances of GP-UCB.

Partial feedback in online learning is not limited to only MAB problems. In fact, the problem of partial monitoring generalizes the idea of MAB to arbitrary feedback functions. Another popular problem is that of label efficient prediction in which additional feedback can be obtained by paying a cost subject to a budget constraint. In fact, label efficient prediction is a special case of the partial monitoring problem. We briefly review these two settings below.

### Partial Monitoring

*Partial Monitoring* provides a general mathematical framework to study the online learning problem under partial feedback. In fact, the other two settings discussed in this section (viz. Label efficient prediction and Multi-Arm Bandits) can be cast as special cases of the partial monitoring problem setting.
A stylized procedure of *Partial Monitoring*:

- **Input:** Loss matrix $L \in N \times M$, Signal matrix $H \in N \times M$
- For $t = 1, 2, \ldots, T$
  - Environment chooses an outcome $o_t \in \{1, \ldots, M\}$
  - Algorithm chooses an action $a_t \in \{1, \ldots, N\}$
  - Algorithm incurs a loss $\ell_{a_t, o_t} = L(a_t, o_t)$ that is not revealed to the algorithm
  - Algorithm is provided a feedback $h_{a_t, o_t} = H(a_t, o_t)$

A finite partial monitoring game $G$ is completely described by two matrices: the loss matrix $L$ and the feedback matrix $H$, each of which is of size $N \times M$. The entries $\ell_{i,j}$ of $L$ are real numbers. The entries $h_{i,j}$ of $H$ are feedback symbols and belong to an alphabet $\Sigma$ (Bartók, Pál, et al. 2011). The $L$ and $H$ matrices are known to the algorithm. In each round, the algorithm chooses an action $a_t \in \{1, \ldots, N\}$ and the environment chooses an outcome $o_t \in \{1, \ldots, M\}$. The algorithm then incurs a loss $\ell_{a_t, o_t}$ and is provided the symbol $h_{a_t, o_t}$. The actual outcome $o_t$ and the loss $\ell_{a_t, o_t}$ are not known to the algorithm. The goal is to minimize the cumulative losses and the regret here is defined against the best single fixed action in hindsight.

### 2.4.2 Label Efficient Prediction

In this setting, the algorithm has a budget (that might be time dependent) on the total feedback signals that can be obtained after each round. That is, after each round $t$, if $\eta(t)$ is the total number of allowed feedback queries and less than $\eta(t)$ queries have been issued so far, then the algorithm can query to obtain the losses of all the possible actions. The goal is keep the losses to a minimum and the regret is measured against the loss of the best fixed action $a^*$.
A stylized procedure of Label Efficient Prediction:

- **Input:** Action set $A$, label space $\mathcal{Y}$, loss function $\ell$, and query rate $\eta : \mathbb{N} \rightarrow \mathbb{N}$

- For $t = 1, 2, \ldots, T$
  - Environment chooses an outcome $y_t \in \mathcal{Y}$
  - Algorithm chooses an action $x_t \in A$
  - Each action $i$ is assigned a loss $\ell(i, y_t)$
  - Algorithm is assigned a loss $\ell(x_t, y_t)$
  - If less than $\eta(t)$ queries have been used so far, the algorithm may query to receive the outcome $y_t$. Else, neither the outcome nor the losses are revealed to the algorithm

The problem was first studied in Helmbold and Panizza (1997) in the restricted setting of $0/1$ classification. In Cesa–Bianchi et al. (2005), a more general setting was tackled with an algorithm that achieves Hannan consistency with high probability.

### 2.5 Summary

We presented a brief overview of online learning under partial feedback. We introduced the concept of online/sequential learning and introduced the important concept of regret which is the standard performance measure in these settings. We also introduced the concept of Hannan consistency, which characterizes effective learning in these settings. We provided particular emphasis to multi–arm bandits since we use this setting extensively in the course of this thesis. We discussed extensions of the basic setting of MAB with more practical and useful variations. We also briefly touched upon other examples of online learning with partial feedback like label efficient prediction and partial monitoring. Finally, we introduced the GP-UCB family of algorithms on which much of the work in this thesis is based upon.
In this chapter, we develop principled methods based on MAB formulations to improve the efficiency of web recommender systems. We first posit the problem of web recommendation systems as a challenge of ranking lists based on click feedback. We present some insights from the XBox recommender system which is a large scale recommendation engine with millions of users and thousands of items. We then propose a technique of efficiently encoding similarities among users and among items. In this problem, the key challenges are threefold: (1) combinatorial number of lists; (2) sparse feedback and (3) context dependent recommendations. We then propose the CGPRANK algorithm, which exploits similarity information specified in terms of a Gaussian process kernel function. CGPRANK can generalize from sparse feedback in three ways: Between positions in a list, between items, and between contexts. We claim that CGPRANK algorithm has strong performance guarantees and show that under natural separability and regularity assumptions, the average regret vanishes. In analyzing the effect of increasing list sizes, we find a surprising theoretical result: With some natural assumptions, increasing the list size leads to faster convergence. We argue that this is due to parallelized exploration instead of an exponential slowdown. We extensively evaluate our approaches on two large datasets from real world recommendation tasks: Firstly, we consider news article recommendation, using data provided by Yahoo! 1. Secondly, we evaluate CGPRANK on Google’s infrastructure, using clickstream data from Google’s ebooks store, demonstrating a significant improvement over existing multi-armed bandit and learning-to-rank techniques.

3.1 INTRODUCTION

In Chapter 1, we provided a brief introduction to the problem of sequentially improving the quality of recommendations. The challenge in

1 http://webscope.sandbox.yahoo.com/
large scale recommender systems is learning from sparse feedback data. However, collecting even these sparse data is not free as bad recommendations lower the quality of the system leading to user abandonment. Traditionally, recommender systems were viewed as a supervised learning problem. This meant that the observed preferences (ratings) of the users were collected into a training dataset, models were trained on this dataset to optimize a metric and finally, the performance was tested on a hold-out test dataset. Existing approaches like collaborative filtering, content based filtering or learning to rank techniques all try to learn a fixed optimal recommendation model given training data. However, these approaches fail to capture the dynamic nature of the user preferences and inventory.

Given a user (or a context or a query), the result of a recommendation interaction is usually a list of items ordered by relevance to the user. An optimal list is one that maximizes a measurable metric (user satisfaction, clicks, purchases, etc.). Learning this optimal ordering of items leads to an “explore–exploit” tradeoff – we need to collect feedback from users about different orderings, while at the same time maximizing the required metric based on the estimated data. Standard multi-arm bandit algorithms which solve such tasks either do not take similarity information into account or cannot select lists of items. On a web scale, this is a daunting task, for mainly three reasons: (1) There is an exponential number of possible lists that we could experiment with; (2) the number of items available for recommendation is often large, compared to the relatively sparse click feedback; (3) optimality of the ordering depends on the context which in itself could be from a large set (e.g., user to whom the item should be recommended).

3.1.1 Related Work

Recommender systems have been studied and deployed since the early days of the internet. Popular techniques for recommendations include collaborative filtering, matrix factorization and frequent item set mining (Ricci et al. 2011), as well as learning to rank approaches (Agichtein et al. 2006; Karatzoglou et al. 2013). These approaches usually estimate user’s preferences (“exploit”) from a fixed training set collected a priori, and generally do not address how to dynamically collect data (“explore”) for training in order to adapt to changing inventories and user bases. A related but important line of work is in information retrieval where the
query and user features together form the context. Incorporating user feedback to modify ranking results has been well studied in the case of product recommendations, web search results (Agichtein et al. 2006), etc. Much of this work again assumes that training data is collected a priori, and do not explicitly address the exploration–exploitation dilemma as we do in this work.

*Top-N recommendations* are an important subclass of recommendation problems (Ricci et al. 2011). Researchers have noted the importance of explore–exploit schemes in dynamic top-N recommendation problems (Lempel 2012) and also found deficiencies in using RMSE optimisation techniques for online recommendations (Cremonesi et al. 2010). We develop efficient techniques for managing the explore-exploit tradeoff and use appropriate regret measures to show a marked improvement over RMSE based schemes.

**Multi-arm bandits (MAB):** As noted in Chapter 2, early approaches such as $\epsilon$-Greedy, UCB1, do not exploit the similarity information between the choices, and thus fail when feedback is sparse or the number of actions is large. Modern research has addressed this challenge, under assumptions of linear (Li, Chu, Langford, and Schapire 2010) and Lipschitz continuous (Kleinberg, Slivkins, et al. 2008) payoff functions. Also, the GP-UCB family described in Chapter 2 attempts to model smoothness of the payoff function with regularity explained by a kernel (Srinivas et al. 2012; Krause and Ong 2011). However, these approaches do not consider the challenges arising when selecting sets and lists. Bandits for subset selection and ranking have been studied before in both ordered and unordered subset selection settings (Koolen et al. 2010). In particular, the best subset selection under ‘bandit-feedback’ setting has been studied in Kale et al. (2010). In Yue and Guestrin (2011); Streeter, Golovin, and Krause (2009), the authors study a similar problem under the setting that the feedback of a set is a submodular function of the concepts covered by the set, which allows to capture diversity, but no similarity among items. Slivkins et al. (2010) considers choosing diverse rankings exploiting item similarity for the problem of ranked document retrieval.
3.2 INSIGHTS FROM A LARGE SCALE RECOMMENDER SYSTEM

We next present some insights gleaned from a large scale recommender system (Microsoft Xbox front dash recommendation). This serves as a motivation for us to develop algorithms that take into account the challenges faced in designing a large scale modern recommendation algorithm. While the actual insights are presented from just one recommender system, we believe that the insights produced and the challenges offered are universal and can be used as a template for any modern web scale recommender system.

Figure 1.2 depicts the Xbox 360 main dash. A list of two personalized items are presented on the upper right hand side. We call this list the main dash list. The goal of this section is to highlight the challenges faced in a list recommendation problem and to study the dynamic nature of the user demand and item interest. One can view these results as cases against using a static model to map context to recommendation list, even if it was learnt from past interaction data. Furthermore, even an algorithm that only updates the learnt model in large batches can perform poorly due to the dynamic nature of the user interest and item fatigue.

3.2.1 Inter-item Similarity Interactions

Recommending a list of size $b$ is typically very different from ranking top-$b$ items. One key difference is the fact that items’ relevance is not independent and diversity/similarity plays a significant role in determining the list’s click probability. In contrast to many previous works that considers an “accuracy vs. diversity trade-off”, we use data from the Xbox recommender system to show that the actual relationship between accuracy and diversity is, in fact, more complex.

Figure 3.1 depicts the click probability items in Xbox’s main dash as a function of both their rating and the Jaccard similarity to the other item presented in the same recommendation list. Sub-figures (a) and (b) depict the empirical click probability surfaces for the items in the upper and lower slots respectively. Sub-figures (c) and (d) depict the predicted probability surfaces for the upper and lower slots respectively.

---

2 The dashboard’s appearance changes with different update versions. This image captures its appearance at the time of data collection.
3.2 Insights from a Large Scale Recommender System

Figure 3.1: Empirical and estimated click probability for items in the upper and lower slots on Xbox’s main dash as a function of the predicted rating and similarity to the other item in the list

using a simple logistic regression model with a third degree polynomial kernel. For the upper slot we see a clear positive correlation between the predicted rating coming from the matrix factorization model (used currently in the Xbox recommender system) and the click probability, however there seems to be very little relevance to the similarity with second item below it. This is not the case with the lower item where a more complex pattern is revealed: For high predicted ratings, it is best to not diversify and recommend “more of the same”, but if the predicted rating is low it is better to “hedge bets” by diversifying the list.

We explain the difference between the upper and lower items by considering the specific layout of the main dash recommendation list (Figure 1.2). The typical user probably notices the upper item before the lower one. Hence, the upper item’s click probability is independent of the lower item but the lower item’s click probability is highly dependent on what is shown above it.

The Xbox recommender is based on a Bayesian model that models the posterior probability of a user to purchase an item (Paquet and
Koenigstein 2013). These probabilities capture both preferences and uncertainty. From Figure 3.1 we learn that if the system has more certainty in user’s preferences it is better to present a list with several similar items even if it means lower diversity. However, when the system’s estimation of the user preference has less certainty, more clicks may be obtained by diversifying the items on the list. Based on these insights, the need to diversify varies and depends on both the predicted rating, the list’s layout and each item’s position.

3.2.2 Item Fatigue

Item fatigue occurs when a user is recommended the same item multiple times. After several impressions of the same item to the same user, the click probability decreases. In Xbox, the recommendation list is refreshed prior to each visit, however some items are repeated multiple times. In the main dash there exists a noticeable effect of item fatigue on the overall CTR. In Figure 3.2, we demonstrate this effect by investigating the variation in the number of impressions before a click event is observed. The data was collected for one month in 2014. For each click event, we counted the number of times the user was presented with that same item prior to clicking on it. We averaged these across all users and items and present the histogram of the number of impressions preceding the first click. The histogram exposes an interesting pattern: The item-fatigue is not always inversely correlated with the click probability. The histogram suggests that there is a threshold number of impressions which is required to maximize the click probability on an item. This threshold varies for different recommender systems and may even change across users and items. This insight
Table 3.1 shows that many users have one or more preferred time-slots in which their consumption of recommendations is considerably higher than in other time-slots.

These insights are presented as a sample of the complexities involved in modern recommender problems. Direct application of existing techniques cannot handle most of these challenges or only handle them through heuristics. Our goal is to develop a principled approach that can also scale to large datasets that are typically encountered.
3.3 PROBLEM SETUP

We have a set of items $V = \{v_1, v_2, \ldots, v_n\}$ (e.g., books, articles etc.) from which the recommendation lists have be generated. We model this task as a sequential decision making problem over $T$ rounds, where, in each round $t$, there is a subset of items $V_t \subseteq V$ available for recommendation (in the default case when all items are available in every round $V_t = V$). Additionally, in each round we receive a context $z_t \in Z$. The context encodes the state of each round and serves as a starting point for the recommendations. Examples of contexts include:

1. The anchor or key item being viewed by the user in this round. This could be the case in an online shopping portal where the user is viewing a particular object and a list of recommendations need to generated based on relevance/similarity to the key item.

2. The user features. This could be the profile information of the user or the preferences/transaction history of the user.

3. The location, time, device used etc.

4. Any combination of the above

While such context could be presented as a feature vector, our algorithm does not require such vectorial representation. Our task is to select an ordered list $L_t = [v_t^{[1]}, \ldots, v_t^{[b]}]$ of $b$ items out of $V_t$ that is to be recommended to the user. In response, we obtain the user feedback as a stochastic vector of rewards, $y_t = [y_t^{[1]}, \ldots, y_t^{[b]}]$, where $E[y_t^{[i]}] = g(v_t^{[i]}, z_t, i)$. We assume that there is some underlying unknown reward function $g : V \times Z \times \{1, \ldots, b\} \rightarrow \mathbb{R}$, such that the expected reward of recommending item $v$ in context $z$ at position $i \in \{1, \ldots, b\}$ is given by $g(v, z, i)$. For concreteness, $g$ may model the click-through rate (CTR), and the rewards $y_t^{[i]} \in [0, 1]$ model whether the user clicks on the item in position $i$. The total reward in round $t$ is thus

$$y_t = \sum_{i=1}^{b} y_t^{[i]} \quad (3.1)$$

and our goal is to maximize the expected cumulative reward $E[\sum_t y_t]$. A crucial assumption we make is that the items in the list do not influence the rewards (clicks) received by other items, i.e., we do not model side
effects of other items in the list. (In the experiments, we do study the
effect of relaxing this assumption) We further assume that the expected
reward
\[ g(v, z, i) = f(v, z) \cdot p(i) \] (3.2)
factorizes into a relevance term \( f(v, z) \) (relevance measures the related-
ness/interestingness of the item to the context which could be a query,
user features or key item) and a position-dependent effect \( p(i) \in [0, 1] \).
Without loss of generality, we assume that \( p(1) = 1 \), and for all \( i \leq j \),
\( p(i) \geq p(j) > 0 \), i.e., showing an item in a later position in the list can
only decrease the expected reward. Under these assumptions, max-
imizing the reward (clicks) is equivalent to positioning the items in
decreasing order of their true relevance \( f \). For a fixed position \( i \) and
context \( z_t \), the expected number of clicks received by an item \( v_{i}^{[i]} \) will be
proportional to its relevance to the context, i.e., \( f(v_{i}^{[i]}, z_t) \). The position
dependence \( p \) can often be estimated effectively (Craswell et al. 2008).
However, the true relevance \( f \) is a priori unknown, and must be esti-
mated through experimentation. We face an exploration–exploitation
dilemma, where we have to choose between exploiting the information
we have about the best ordering, and exploring alternate orderings,
which may or may not lead to higher rewards. Instead of maximizing
rewards, in the following we equivalently wish to minimize the regret.
Hereby, the instantaneous regret \( r_t \) in round \( t \) is given by:
\[ r_t = \sum_{i=1}^{b} [g(v_{i}^{[i]}*, z_t, i) - g(v_{i}^{[i]}, z_t, i)] \] (3.3)
where \( \mathcal{L}_t^* = [v_{t}^{[1]*}, \ldots, v_{t}^{[b]*}] \) is an optimal (in expectation) ordered list
for context \( z_t \) observed at round \( t \). Our goal then is to minimize the
cumulative regret, \( \mathcal{R}_t = \sum_{t=1}^{T} r_t \). In particular, we desire an algorithm
such that achieves Hanan consistency i.e., \( \mathcal{R}_t / T \to 0 \) as \( T \to \infty \). Note
that this is quite a stringent performance requirement: Vanishing regret
requires that the algorithm learns the optimal (in expectation) mapping
from context to recommendations. Regret is a better measure of per-
formance than the traditionally used root mean square error (RMSE).
This is because RMSE is measured over all (context, item) pairs and
overall reduction of RMSE could be achieved by reducing the error in
prediction of (context, item) pairs that would never occur in practice.
3.4 Algorithm and Analysis

Given the formulation, we address the resulting challenges in this section. In Section 3.4.1, we show how the item feedback can be shared across positions. In Section 3.4.2, we tackle the problem of principled generalization of feedback to items/contexts that are not yet explored using appropriate statistical models. Finally, the resulting exploration–exploitation dilemma is addressed in Section 3.4.3 leading up to the specification of the CGPRANK algorithm. Before we describe the algorithm in Section 3.4.4 and provide theoretical guarantees in Section 3.4.5, we highlight again the key technical challenges:

1. There is a combinatorial number of possible ordered lists. When \( n \) is large, these exponentially many choices are intractable even for small \( b \). In Section 3.4.1, we show how we can share item feedback across positions in order to reduce this complexity.

2. In many applications, click feedback is sparse, potentially severely delaying convergence. In Section 3.4.2, we discuss the use of statistical models for generalizing the feedback to items that are not yet explored in a principled way.

3. Once we have settled on a statistical model for learning about reward, we face the exploration–exploitation dilemma of trading experimentation (for the purpose of parameter estimation) and exploitation (using the model’s predictions to maximize reward). This dilemma is addressed in Section 3.4.3.

3.4.1 Sharing feedback across positions

Given a context, selecting an optimal ranked list of \( b \) recommendations is challenging due to the combinatorial number of choices. In the worst case, we may need to estimate the reward associated with each of the exponentially many rankings. However, under our assumptions (3.1) and (3.2) that the reward of a list decomposes additively over the positions, and that the position-dependent reward factors into a position-dependent effect independent of the item and a “relevance” effect that is position independent, the problem becomes statistically and computationally more tractable. If we know the position effects \( p(i), \forall i \in \{1, ..., b\} \), we can normalize the feedback received by an item across all positions that it has been shown at so far. That is, given
context $z_t$, if we observe feedback $y_{i_t}^{[i]}$ for some item $v$ shown in position $i$, $y_{i_t}^{[i]} / p(i)$ provides an unbiased estimate of $f(v, z_t)$. Consequently, an unbiased estimate for the reward obtained when showing $v$ in position $j$ instead is given by $y_{i_t}^{[i]} \cdot p(j) / p(i)$. This insight thus allows us to share feedback across positions.

### 3.4.2 Sharing across items/contexts via kernels

In order to generalize feedback across items/contexts, we need to incorporate prior information about their respective similarities. We assume that this prior information is presented in terms of an arbitrary positive definite kernel function $k : (V \times Z)^2 \rightarrow \mathbb{R}$. Hereby, for two item-context pairs $(v, z)$ and $(v', z')$, the kernel $k((v, z), (v', z'))$ represents our assumptions about how similar we expect the rewards to be when presenting item $v$ in context $z$, as opposed to presenting item $v'$ in context $z'$. A multitude of kernel functions are available for accurately capturing similarity among various types of data (Schölkopf and Smola 2001). When we choose a particular kernel, we effectively assume the reward function $f$ can be represented as a linear combination

$$f(v, z) = \sum_j \alpha_j k((v, z), (v_j, z_j)),$$

i.e., as a basis function expansion around a set of context–item pairs $((v_j, z_j))$. Such functions span the Reproducing Kernel Hilbert Space (RKHS) associated with kernel $k$, and the norm of $f$ in that space,

$$||f||_k = \sum_{i,j} \alpha_i \alpha_j k((v_i, z_i), (v_j, z_j)),$$

measures the “complexity” (regularity) of function $f$. The performance of our algorithm, as analyzed in Theorem 3.1, will depend on this norm. Intuitively, if the kernel matches the regularity present in real data well, the norm will be small. Capturing similarity via kernels has important consequences: In particular, it allows interpreting the relevance function $f$ as a sample from a Gaussian Process (GP) prior (Rasmussen and Williams 2005), with covariance (or kernel) function $\kappa$. Consequently, one interprets the relevance as a collection of normally distributed random variables, one for each item—context pair. They are
jointly distributed in a dependent manner, such that their covariances are given by the kernel:

$$\text{Cov}(f(v,z), f(v_j,z_j)) = k((v,z), (v_j,z_j)).$$

This joint distribution then allows us to make predictions about unobserved item–context pairs via inference in the GP model. In particular, suppose we have already observed feedback for $\bar{y} = [y_1, \ldots, y_t]$ from $t$ recommendations, i.e., obtained data $D = \{(v_1, z_1, y_1), \ldots, (v_t, z_t, y_t)\}$. Then, for a new item–context pair $(v,z)$, its predictive distribution for $f(v,z)$ is Gaussian, with mean and variance given by:

$$\mu_t(v,z) = k_t(v,z)^T(K_t + I)^{-1}\bar{y},$$

$$\sigma^2_t(v,z) = k((v,z),(v,z)) - k_t(v,z)^T(K_t + I)^{-1}k_t(v,z),$$

where $k_t(v,z) = [k((v_1, z_1), (v,z)), \ldots, k((v_t, z_t), (v,z))]^T$ and $K_t$ is the positive semi-definite kernel matrix such that $K_{i,j} = [k((v_i, z_i), (v_j, z_j))]$.

**Choice of Kernels.** Often, kernels over item–context pairs are naturally expressed as tensor products, where

$$k((v,z), (v',z')) = k_V(v,v') \cdot k_Z(z,z').$$

where $k_V : V^2 \to \mathbb{R}$ is a kernel among items, and $k_Z : Z^2 \to \mathbb{R}$ is a kernel among contexts. This choice of kernel expresses our prior assumption of how smoothly the CTR changes over the item-context space.

The choice of kernel $k_V$ depends on the particular recommendation problem. Often, similarity between items is given by a usually symmetric similarity function $\text{sim} : V^2 \to R$. A valid kernel function however must additionally be positive definite (i.e., all resulting covariance matrices must be positive definite). Among various available candidates, we use diffusion kernels, a family of kernels first introduced in Kondor and Lafferty (2002).

The first step is to consider the items $V$ as nodes in a weighted, undirected graph $G$, so that the weight $w(i,j)$ of each edge $(i,j)$ is given by the similarity function $\text{sim}(i,j)$. The diffusion kernel is then given as matrix-exponential $K_V = \exp(\alpha L)$ of the graph Laplacian $L$ of $G$.

In the contextual setting, if the context is given as a key item, the same diffusion kernel can be used both for items and contexts. If the
context is given in terms of user features, \( k_Z \) can be chosen, e.g., as linear
\( k_Z(z, z') = z^T z' \), or Gaussian \( k_Z(z, z') = \exp(-\frac{1}{2}\|z - z'\|^2/h^2) \). If no
similarity information is known between contexts, the diagonal kernel
\( k_Z(z, z') = 1_{[z = z']} \) can be used. When features are explicitly available,
we can use linear kernels, other kernels defined over Euclidean spaces
or combinations thereof. In a special case of CGPRank, we can recover
the exact algorithms presented in Chu, Li, et al. (2011) and
Li, Chu, Langford, and Schapire (2010) by choosing appropriate linear
kernels. In our experiments, we employ CGPRank with both diffusion
and linear kernels and demonstrate their performance. However, in
several real world applications, features are not easily available either
for the contexts or the items and the nature of CGPRank allows us
to use any kind of kernel that can be computed from the similarity
information that is available.

3.4.3 Explore-Exploit in List Selection

While the predictive model (Equations (3.4) and (3.5)) allow us to update
our models based on feedback, we still need to pick an ordering of
items in each round. One approach could be to greedily maximize the
expected reward according to our current model (i.e., rank items in
order of their predictive mean (3.4)). However, this approach ignores
the predictive uncertainty (3.5). If our goal were to conduct experiments
to most effectively reduce uncertainty about the model, we may instead
consider to pick items according to the predictive variance (3.5). Note
that this would be the uncertainty sampling procedure popular in active
learning settings. Such an approach however would incur high regret,
since it would equally explore high- and low value items. Therefore, in
each step, we must trade off experimentation (showing items we have
not explored yet) and exploitation (showing items with high expected
reward). One way to achieve this is linearly trade off the relative
importance of the predictive mean and the predictive variance to score
each candidate item. Here we could use the sampling procedure of
GP-UCB introduced in Chapter 2. This would mean selecting the item
\( v \) that maximizes, for the current context \( z_t \) the surrogate objective
\( UCB_{v, z_t} \), where

\[
UCB_{v, z_t} = \mu_{t-1}(v, z_t) + \beta^{1/2}_{t,v} \sigma_{t-1}(v, z_t).
\]
For Gaussian predictive distributions, this criterion captures an upper confidence bound (UCB), i.e., an upper bound on the relevance function that holds with a certain probability that can be controlled via the tradeoff factor $\beta_{t,v}$. That is, the respective weighting of the mean and variance is handled by an item and time dependent variable, $\beta_{t,v}$. We show how to pick $\beta_{t,v}$ in Section 3.4.4 such that, with high probability, the UCB provides a valid upper bound on the true mean. At the same time, the choice is small enough so that the instantaneous regret provably decreases quickly over iterations.

In order to pick multiple items in each round, a first attempt would be to score every item $v$ according to the selection rule (3.6), and select the $b$ highest scoring items. However, given the regularity imposed by the kernel function, for a fixed context $z$, the highest scoring items are likely very similar. Thus, the resulting list will explore sets of highly related items together, in a possibly redundant manner.

Instead, it may be desirable to encourage diversity when selecting lists to explore. One natural, and computationally efficient way is to anticipate the reduction in uncertainty achieved by the items already selected. Recall from Chapter 2 that this is the intuition behind GP-BUCB proposed by Desautels et al. (2014) for parallelizing bandit optimization problem. Looking at the predictive mean and variance $\mu_t$ and $\sigma_t$ in Equations (3.4) and (3.5), it can be observed that, while $\mu_t$ depends on the actual feedback $\bar{y}$ observed so far, the predictive variance $\sigma_t^2$ does not depend on previous feedback. We can utilize this insight in the following way. Suppose, in round $t$, we receive context $z_t$, and wish to recommend a list $\mathcal{L}_t = [v_t^1, \ldots, v_t^b]$. We select the first item $v_t^1$ according to (3.6). Then, we update the predictive variance (3.5) as if we had already observed the feedback for the first item. The predictive mean is not updated (or equivalently, it is updated with its own prediction). Note that this will have the effect that the predictive variance – and hence the score (3.6) – for similar items is decreased. We now select the second item $v_t^2$ according to the updated score, and proceed in this manner until the full list of $b$ items has been selected.

After the ranked list has been selected, feedback $y_t = [y_t^1, \ldots, y_t^b]$ is observed. According to our factorization assumption (3.2), each observation $y_t^i$ in position $i$ provides a noisy observation of the underlying relevance score $\mathbb{E}[y_t^i] = p(i) \cdot f(v_t^i, z_t)$. Hence, we provide as feedback $\hat{y}_t^i / p(i)$ as unbiased estimate of $f(v_t^i, z_t)$. 

40
3.4.4 Computing the Tradeoff Parameter

We now describe how to compute a value for $\beta_{t,v}$ that allows us to prove rigorous bounds on the regret of CGPRank. Note that in practice a more aggressive choice than this conservative prescription can lead to faster convergence. CGPRank extends and generalizes the work of Srinivas et al. (2012); Krause and Ong (2011); Desautels et al. (2014). In these, the tradeoff parameter $\beta_t$ ensures that, in each iteration, the true relevance function is contained within the constructed confidence bands $(\mu_t(v) \pm \beta_t \sigma_t(v))$ with high probability. Similarly, for our problem, we compute $\beta_{t,v}$ as

$$\beta_{t,v} = C'_b \left( 2M^2 + 300\ln^3 \left( \frac{tb}{\sigma} \right) \left( C_t + \frac{1}{2} \log \left( 1 + \sigma^{-2} \sigma_{t-1}^2(v,z_t) \right) \right) \right)$$  

(3.7)

where $C'_b = \frac{1}{p(b)^2} \max_{L \subseteq V \times \{z_t\} : |L| = b} \left| \frac{1}{2} \left( \mathbb{I} + \sigma^{-2} K_{t-1}(L,L) \right) \right|$ and $C_t = \frac{1}{2} \sum_{\tau=1}^{t-1} \sum_{i=1}^{b} \log \left( 1 + \sigma^{-2} \sigma_{t-1,i}^2(v^{[i]}_\tau, z_\tau) \right)$.

Hereby $M$ is a bound on the RKHS norm of the reward function $f$, and $\sigma_{t-1,i}^2(v^{[i]}_\tau, z_\tau)$ is the predictive variance after having selected items 1 to $i-1$ in iteration $\tau$. Note that $C_t$ can be computed efficiently incrementally over the course of the algorithm. $C'_b$ depends on the maximum determinant of any (posterior) kernel matrix $K_{t-1}(L,L)$ that can be constructed using at most $b$ items paired with the current context. While computing this quantity exactly requires solving a combinatorial optimization problem, it can be approximated efficiently and accurately during each iteration by running a simple greedy algorithm (uncertainty sampling). For several commonly used kernel functions (linear, Gaussian and combinations thereof), $\beta_{t,v}$ can be tightly bounded by the simple expression $\beta_t = C \log^d t$ with suitable constants $C, d'$. It is this form that we use in the experiments. Algorithm 3.1 presents pseudocode for our CGPRank algorithm. The procedure GP-Inference$(k,D)$ takes a kernel function $k$ and data set $D$, and returns the predictive mean and variance functions according to (3.4) and (3.5).
3.4.5 Regret Analysis of CGPrank

Our analysis builds on and extends results of Krause and Ong (2011) for contextual GP bandit optimization (selecting individual items) and Desautels et al. (2014) for non-contextual GP bandit optimization with delayed feedback. We state our main result in the form of the following theorem.

**Theorem 3.1.** Let $\delta \in (0, 1)$, $M > 0$, and $k$ be a kernel function, such that $\|f\|_k \leq M$. In each round, choosing $\beta_{t,v}$ as specified in Equation 3.7 and running CGPrank for $T$ rounds, it holds that

$$\Pr\{\mathcal{R}_t \leq \frac{\sqrt{Tb\gamma_Tb(C_1 + C_2\gamma_Tb \ln^3(Tb/\delta))}}{\gamma_Tb} \forall T \geq 1\} \geq 1 - \delta$$

where, $C_1 = \frac{16\exp(2\gamma_b)M^2}{\log(1 + \sigma^{-2})}, C_2 = 300$

and $\gamma_n = \max_{\mathcal{D} \subseteq V \times Z, |\mathcal{D}| \leq n} \log |I + \sigma^{-2}K(D, D)|$

The regret bound in Theorem 3.1 depends on the quantity $\gamma_n$, which quantifies the effective degrees of freedom of the kernel matrix $K(D, D)$ that can be constructed from $n$ context-item pairs. This quantity was analyzed in prior work (Srinivas et al. 2012), showing that for many common kernels (such as linear and Gaussian), $\gamma_n$ only grows polylogarithmically in $n$. The most instructive way to interpret Theorem 3.1 is to consider the average regret per list slot, $\mathcal{R}_t / Tb$. We can infer that, as long as $\gamma_n$ grows only polylogarithmically in $n$ (the common case),

$$\frac{\mathcal{R}_t}{Tb} = O\left(\frac{p(1)}{p(b)}\sqrt{\frac{\gamma_TbM^2\exp(2\gamma_b)}{Tb}}\right) = O^*\left(\frac{M}{p(b)}\sqrt{\frac{\exp(2\gamma_b)}{Tb}}\right),$$

where the $O^*$ notation hides logarithmic factors in $T$ and $b$. Thus, for fixed list size $b$, the average regret per slot decays to 0 at an essential rate of $O^*(\frac{1}{\sqrt{T}})$. It grows linearly with the complexity $M$ of the reward function $f$, and inversely proportional to the decay of the position effect $p(b)$.

How does the regret scale with list size? Since $\exp(\gamma_b) = \Omega(b)$, as the list size $b$ increases, straightforward application of the algorithm will incur average regret per slot that increases with $b$. However, in the non-contextual case (or the case of a finite set of contexts), it is possible
to slightly modify the algorithm, such that, as long as $b = O(\log T)$, it can be ensured that $\gamma_b$ remains bounded \textit{irrespective of} $b$, at the cost of additional regret bounded by $O(\text{poly log}(T))$. Thus, in this setting, one can achieve an average regret per slot of

$$\frac{R_t}{Tb} = O^\ast \left( \frac{M}{p(b)} \sqrt{\frac{1}{Tb}} \right) \quad (3.8)$$

This result suggests that, perhaps surprisingly, as long as $p(b) \geq \frac{1}{\sqrt{b}}$, increasing the list size can lead to faster convergence. This finding is further supported by our experimental results in Section 3.7. We now present a proof sketch of our claims.

\textbf{Proof sketch of Theorem 3.1:} We build on Theorem 1 of Krause and Ong (2011) and Theorem 1 of Desautels et al. (2014). Consider $\alpha_t = (2M^2 + 300\gamma t b \ln^3 t b / \delta)$. From Krause and Ong (2011), we know that if we apply a fully sequential, hypothetical variant of CGPRANK, which, after selecting an item to add to the list, immediately obtains feedback, and uses $\alpha_t$ instead of $\beta_t$ in its selection rule guarantees that

$$\Pr \{ R_t \leq \sqrt{C_1 Tb \alpha T \gamma Tb} \forall T \geq 1 \} \geq 1 - \delta. \quad (3.9)$$

To prove this claim, we first observe that, while greedily assembling a list $L_t = [v_t^1, \ldots, v_t^b]$ one element at a time, it follows from Theorem 4 from Srinivas et al. (2012), that, setting $\sigma = 1/p(b)$, the true relatedness $f(\cdot)$ is contained within the confidence bands $\mu(\cdot) \pm \sqrt{\beta_t} \sigma(\cdot)$. Thus, from (a slight generalization of) Lemma 4.1. of Krause and Ong (2011), it follows that, in step $i$, the instantaneous regret of selecting $v_t[i]$ as compared to $v_t[i]^*$ is bounded by $\sqrt{\beta_t} \sigma(\cdot)$. Claim (3.9) then follows from Theorem 5 of Krause and Ong (2011).

We now interpret CGPRANK as implementing the hypothetical, fully sequential algorithm above with delayed feedback (i.e., feedback is only obtained after an entire list is recommended). The effect of delayed feedback is studied by Desautels et al. (2014). They prove in Theorem 1 that as long as $\beta_t$ is chosen as $\beta_t = \alpha_t \exp(2C)$, where $C$ is a bound on the maximum mutual information than can be possibly obtained while feedback is delayed, satisfies the regret bounds

$$\Pr \{ R_t \leq \sqrt{C_1 Tb \beta T \gamma Tb} \forall T \geq 1 \} \geq 1 - \delta. \quad (3.10)$$

While their result holds for the non-contextual setting, it can be generalized in a straightforward manner to the contextual setting, as long
Algorithm 3.1 The CGPRank algorithm

**Input:** Kernel \( k \), selection batch size, \( b \)
Initialize data set of observations \( D = {} \).

for \( t = 1, 2, \ldots, T \) do
    Observe context \( z_t \in Z \)
    Receive set of available items \( V_t \)
    Set \( \hat{D} \leftarrow D \)
    for \( i = 1, 2, \ldots, b \) do
        \( [\mu(\cdot), \sigma^2(\cdot)] \leftarrow \text{GP-Inference}(k, \hat{D}) \)
        \( v_t^{[i]} \leftarrow \text{argmax}_{v \in V_t} \mu(v, z_t) + \beta_1^{1/2} \sigma(v, z_t) \)
        \( \hat{D} \leftarrow \hat{D} \cup \{(v_t^{[i]}, z_t, \mu(v_t^{[i]}, z_t))\} \)
    end for
    Recommend list \( L_t = [v_t^{[1]}, \ldots, v_t^{[b]}] \)
    Observe feedback \( y_t = [y_t^{[1]}, \ldots, y_t^{[b]}] \)
    \( D \leftarrow D \cup \{(v_t^{[1]}, z_t, y_t^{[1]}/p(1)), \ldots, (v_t^{[b]}, z_t, y_t^{[b]}/p(b))\} \)
end for

as the context remains constant until the actual feedback is obtained (which is the case in our setting).

Note that \( C \leq \gamma_b \) by definition of the capacity \( \gamma \). Further, note that as shown in Theorem 2 of Desautels et al. (2014), it is possible to bound \( \gamma_b \) by a constant independent of \( b = O(\log t) \), by applying an explicit pure-exploration phase of length \( O(\text{poly log } T) \). This same technique can be applied to initialize CGPRank, at least in the non-contextual case (\( |Z| = 1 \)). It can also be applied in case of a finite number of contexts, in which case active exploration has to be applied separately in each context. This proves the claim of Equation (3.8).

3.5 Scaling to Web Scale Recommendation Tasks

Naively implementing Algorithm 3.1 can be prohibitively slow for large data sets. For general kernels, the data set size \( D \) grows with the number of observations \( Tb \), and performing exact Bayesian inference according to Equations (3.4) and (3.5) requires solving linear systems in \( Tb \) variables.
• **Scaling GP Inference.** Fortunately, much work has been done scaling GP inference to massive data sets (Rasmussen and Williams 2005), also in online/streaming settings (Gomes and Krause 2010). Since such inference is the essential subroutine in CGPRank, it can immediately benefit from these techniques. Furthermore, in many practical applications (such as the recommendation tasks considered in our experiments), the kernel $k$ is of bounded rank $d$, in which case inference only requires solving a linear system in $d$ dimensions. Often, approximate solutions are acceptable for practical performance.

• **Speeding up selection:** In order to speed up the selection rule (4.4), another computational trick can dramatically accelerate performance. Note that, in order to evaluate (4.4) naively, the mean $\mu_{t-1}(v, z_t)$ and variance $\sigma_{t-1}^2(v, z_t)$ has to be computed for each choice of $v \in V_t$. Inspecting Equations (3.4) and (3.5), it can be seen that computing (3.4) requires solving only one linear system, while computing (3.5) requires solving $|V_t|$ linear systems. By exploiting the fact that, in GPs, predictive variance must monotonically decrease, i.e., $\sigma_T^2(v, z) \geq \sigma_{T+1}^2(v, z)$, previous estimates can be used as upper bounds. This insight allows to use priority queues to dramatically reduce the number of linear systems that need to be solved. Similar ideas have been exploited in Desautels et al. (2014).

• **Delaying feedback:** Instead of continuously performing updates, CGPRank can be accelerated by reusing the same recommendation multiple times, accumulating feedback and performing delayed updates. However, delaying feedback for long periods of time can incur higher regret in the intermediate period where a fixed suboptimal ordering is chosen. Hence, careful choice of the frequency depending on the problem domain and taking into consideration speed of accumulation of feedback is an important aspect of scaling up CGPRank.

Exploiting some of these techniques, in our experiments with the Yahoo! dataset and using linear kernels, we were able to achieve an average selection time of 0.4 millisecond per slot including updating the model based on non-delayed feedback (timed on unoptimized C++
code compiled using gnu compiler and running on a single core of a Quad Core Intel Xeon E3, 3.5GHZ machine with 32GB RAM)

### 3.6 Experimental Setup

We extensively evaluate CGPRANK on two real-world recommendation tasks. The following questions guide our experimental study:

1. Can we exploit similarity to achieve accelerated convergence?
2. Can one parallelize exploration across lists to achieve faster convergence?
3. Can improved performance be achieved by incorporating context?

**Benchmarks.** In our experiments, we use the following approaches and compare the performance:

- **CGPRANK-Lin** is the version of CGPRANK that uses linear kernels to model similarity between articles and between contexts. This version of the algorithm is similar to LinUCB-HYBRID algorithm which was originally designed for single-item selection discussed in Li, Chu, Langford, and Schapire (2010). In fact, LinUCB-HYBRID is a special case of CGPRANK.
- **CGPRANK-G**, is the version of CGPRANK with graph kernels on the items and clustering of contexts to model similarity. and can be used when we do not have access to user or item features. We use this version of CGPRANK for the experiments on Google books data.
- **CGPRANK-b-Lin**, which simply selects the top $b$ items according to score (4.4)
- **UCB1** of Auer, Cesa–Bianchi, and Fischer (2002). For list selection, we pick the top $b$ items according to the UCB score. We also use a clustered version where we maintain an independent instance of the algorithm per cluster of contexts.
- Hierarchical versions of both UCB1 and CGPRANK (RANK-UCB and RANK-LinUCB), based on Yue and Guestrin (2011); Streeter, Golovin, and Krause (2009).
- Learning to Rank Approaches: These are non-adaptive baselines that use a fraction of the data to train the model and then provide
3.6 EXPERIMENTAL SETUP

a ranking for every user request. In particular, we compare against two best performing algorithms from the RankLib module of the Lemur Toolkit software \(^4\) (Coordinate Ascent (LTR-CA) (Metzler and Bruce Croft 2007) and Rank Boost (LTR-RB) (Freund et al. 2003))

- **Random** selection of lists.
- **Hindsight-Fixed** selection: Picking lists that are optimized in hindsight. This is an (unrealistic) upper bound benchmark.

![Graph](image)

(a) Article selection on Yahoo! News  
(b) Performance on Google logs

**Figure 3.3**: a: Total clicks received by the algorithms when employed to select a single article per round, using unbiased estimates from log data. b: Results of using CGPRank-G to rerank recommendations on a clickstream log from Google’s ebooks store. For confidentiality, we only present normalized numbers. Note that CGPRank-G offers a significant (~18%) improvement over the existing non-adaptive method and also outperforms top-b UCB1.

### 3.6.1 Yahoo! news article recommendation

**Data set**: We first evaluate our algorithm on clickstream data made available by Yahoo! as part of the Yahoo! Webscope program (Yahoo! [4](http://http://www.lemurproject.org/)

---

Figure 3.4: a: Contextual list selection task with $b = 4$. These results are based on click feedback simulated according to the logs. b: Similar to 3.4a. Contextual list selection task with $b = 4$.

Figure 3.5: a Effect of $p(b)$ on the final total regret demonstrated for a $b = 2$ item selection task. The regret increases as $p(2)$ decreases, but not dramatically so. b demonstrates the power of parallelizing exploration within lists to achieve accelerated convergence. In terms of the per-slot average regret, it shows how the relative performance compared to Ideal – perhaps surprisingly – improves with increasing $b$ (also compare Section 3.4.5).
3.6 Experimental Setup

Webscope™ Program 2013). Specifically, we use the R6A dataset containing a part of the user view/click log for articles displayed on the Today Module of Yahoo! during ten contiguous days. This data was collected in May 2009 and the displayed article was chosen uniformly at random from the list of available articles. This makes this dataset ideal for unbiased, offline evaluation of exploration–exploitation approaches. One can find detailed information on the dataset, the data collection methodology and an explanation of the unbiased offline evaluation in Yahoo! Webscope™ Program (2013); Li, Chu, Langford, and Xuanhui Wang (2011); Chu, Park, et al. (2009). The dataset consists of more than 45 million lines of log. Each line consists of the following information:

- The timestamp of the user visit.
- The article ID of the actual displayed article and whether a click was recorded or not.
- Anonymized user features denoting the context
- Article features for all the articles that were available in the pool for selection

Parameter choice: We initially chose the first 10% of the log entries as initialization data for optimizing parameters of the algorithms, training the learning-to-rank methods, and also to extract user features for clustering (see below). We used the results of this clustering for the contextual versions of CGPRANK-G and UCB1. We ignored this part of the data for all further evaluations the results reported are completely evaluated on the remaining 90% of the log entries.

The hindsight-fixed benchmark for these experiments used a weight vector obtained by solving to a linear regression problem for the click prediction task based on the entire log. The result is a single weight vector that maps any item-context pair \((v, z)\) to an expected click probability.

The nature of the data set makes linear kernels the ideal choice for this task. With this choice, CGPRANK-Lin for single-item selection corresponds to the LINUCB-HYBRID algorithm as provided in Li, Chu, Langford, and Schapire (2010). In order to use CGPRANK-G on this dataset, we require a kernel function on the articles. We decided to model the articles as nodes in a graph. The weight on the edge connecting any two articles is simply the euclidean distance between
their feature vectors. This choice allows us to compute a diffusion kernel on the articles. Note that computation of an appropriate diffusion kernel requires tuning of the heat parameter $\alpha$. For our evaluations, we used the article features and their corresponding clickthrough rates to tune the parameter alpha $\alpha$ on the first 10% of the data.

For contextual versions of CGPrank-G and UCB1, we used a simple technique of clustering the user features given in the logs, and maintaining one instance of all the evaluated algorithms per cluster (corresponding to a diagonal kernel $k_Z$). We used $k$-means clustering on the user contexts extracted from the initialization bucket with number of clusters, $k = 10$, picking the best solution from multiple random restarts. During the actual evaluation, in each round $t$, the user context given in the log line was mapped to its nearest cluster center, and $z_t$ was set to this cluster index.

We carry out experiments both in the contextual and non-contextual setting, and vary the size of the lists selected. For list size $b = 1$, we use the actual click feedback given by the log. For $b > 1$, we use simulated click feedback as described above. In order to maintain consistency over the amount of feedback available, we randomly sample portions of the actual log during our simulated feedback since the rejection sampling technique used for $b = 1$ provides feedback once in 20 iterations in expectation.

**Feedback.** While the goal of our work was to choose the optimal ordered list for recommendation, this dataset only contains click stream data for the choice of a single item. Hence, for the purpose of evaluating the list selection procedures, we simulated list feedback. The feedback for an article at a given position depended on the base Clickthrough rate (CTR hereafter) of the article and the bias introduced by the position. Hence, given context $z$, if an article $v$ with base CTR $f(v, z)$ was shown at position $j$ with a bias of $p(j)$, then the stochastic feedback for this placement was simulated as a Bernoulli draw with click probability $f(v, z)p(j)$. Estimating the positional effects $p(j)$ on CTR is a well studied problem. The base CTR $f(v, z)$ used in the simulation was computed as the CTR predicted by the hindsight-fixed algorithm for the given $(v, z)$ pair.
3.6.2 *Google e-books recommendation*

**Data set.** We carry out our second set of experiments on clickstream logs from the Google ebooks store. Here, the recommendation task is, given a key book (context) the user is currently exploring, recommend a set of related books that the user may also be interested in.

At the time of this work, Google used metadata information about the books and also inputs from other sources to compute the ordering of the related list of books to any given key book. This is a good first approximation of true “relatedness” in the absence of any real click data. But, as we receive feedback in terms of clicks on the recommendations, we can modify the original ordering in order to reflect the tastes of the users and this new ordering represents the true “relatedness” of books in the presence of a large number of clicks. In this dataset, the only context available was the current item being viewed (key item).

We evaluated our algorithm on the clicklog data of Google’s book store that was collected over 42 days in the beginning of 2012. Each event in the anonymized click log data consists of two components:

- The volume id, identifying the key book (anchor item);
- The position of the related book on which the user clicked in the related list.

We estimated the unbiased position effect on the CTR using standard techniques.

**Parameter choice.** For each key book \( z \), we created a graph structure capturing the initial ordering given by the metadata-similarity in terms of the edge weights. Because of computational considerations, we only consider the similarity between the key book \( z_t \) and all of its candidate books \( V_t \), but not the similarities between the candidate books themselves. This results in a star graph with the key book in the center. The weights on the edges are the similarity scores between the books as computed using the metadata of the books. Using the obtained related graph \( G \), the diffusion kernel \( K \) can be computed using techniques presented in Kondor and Lafferty (2002).

**Feedback.** Based on the data, we simulate feedback for each item when it was displayed in a specific related list. Note that the clicks are being aggregated over users and sessions such that we group feedback on a specific related list. Position independent base CTR models how much the users prefer seeing a related book \( v \) in the recommendation
list of key item \( z \). We define this CTR as the number of position-normalized clicks that item \( v \) got while being shown in the related list for key book \( z \), divided by the position-normalized number of times \( v \) was shown as a related book for \( z \).

Based on these estimates, we use offline evaluation techniques to simulate feedback for any new ordering. Since we computed the position independent feedback for each of the items in the original list and we also have the position weight terms \( p(j) \), given context \( z \), we simulate feedback for any item \( v \) with base CTR \( f(v, z) \) at position \( j \) with position weight \( p(j) \) by sampling from a Bernoulli distribution with bias \( f(v, z)p(j) \).

3.7 Results

**Performance comparison.** The results on the Yahoo! webscope dataset presented in Figure 3.3a and the results from the Google books evaluation presented in Figure 3.3b show that all versions of CGPRANK offers a consistent performance improvement over approaches that do not take item similarity into account. The ability of CGPRANK to generalize feedback received from few items to a larger set of related items allows it to quickly estimate their relevance and converge towards optimal selections earlier. Also, in a dynamic system where new items regularly become available for selection, this feature of CGPRANK allows it to decrease the number of times the new item has to be chosen before its relevance is reliably estimated. Thus, CGPRANK is well suited to handle the cold start problem in recommendations. For the Yahoo! Webscope dataset, CGPRANK produced an overall final CTR of 0.0496 for the context-free setting and 0.0603 for the contextual setting, which compares favorably with the Ideal policy (0.0559 and 0.064). In the case of the Google books dataset, CGPRANK outperforms the then-existing algorithm by a margin of 18%, despite having to handle a large number of items. These findings substantiate our first hypothesis, that sharing feedback across similar items helps.

**Performance without features.** We decided to further test the performance of CGPRANK in settings where no explicit features are available. Instead, similarity information was provided in the form of a kernel function and the similarity between contexts was taken into account by clustering. The evaluation of this is also presented in Figures 3.3a and 3.4b. While the overall performance of CGPRANK-G was outperformed
by CGPrank-Lin (CTR of 0.0574 compared to 0.0603 for single item selection), it still manages to perform well and is applicable even when explicit features are not present.

In Figures 3.4a and 3.4b present the results of the list selection task with \( b = 4 \). From the plots, it can be inferred that adaptively learning the order is better than any fixed model learnt from training data. It can also be seen that even using context in an arguably naive manner (in terms of clustering users) provides a substantial improvement over not using context. For the single article selection case with actual click feedback, there was a 14% increase in the CTR while utilizing context information, further substantiating our hypothesis that exploiting context helps.

Relaxing the independence assumption. CGPrank assumes that items do not influence the feedback of other items within a list. This simplifies the algorithm and its analysis, but is not necessarily true in practice. Hence, we relax this assumption by clustering the articles and model the user as diversity-preferring by ensuring that at most one article from a cluster is clicked on in a round. The recommended list might still contain multiple items from the same cluster. Although the total regret is \( \sim 5\% \) more than in the independent case, CGPrank still outperforms all other baselines and is better than the next best baseline by \( \sim 10\% \).

Parallelizing exploration within lists. In our analysis in Section 3.4.5, we found that, perhaps surprisingly, increasing the list size can lead to accelerated convergence – at least under certain technical assumptions – as exploration is “parallelized” across list slots. We empirically assess this finding in Figure 3.5b which considers the per-slot average regret. In this experiment, we apply CGPrank on the log data, using different batch sizes \( b \). As \( b \) increases, faster convergence is obtained in relative terms compared to the hindsight-fixed predictions.

The experiments with multiple item selection corroborate our theoretical claims that having to select multiple items is beneficial if we consider the per slot regret as long as we gather enough feedback in the lower ranked slots. From the figures, it can be seen that while average regret per slot decreases as we move from single item selection to 2 and then 4 items, there is diminishing returns when we select 8 items. This is due to the low position CTR at positions higher than 4 and also the sparse nature of feedback in the problem. As long as \( p(i) \) is high enough to garner enough feedback, the opportunity cost incurred by
making poorer recommendations down the order is less than that at
the top of the list. This is because of the decreasing expected CTR \( p(i) \)
as position \( i \) increases. During the actual execution, it can be noticed
that CGPRANK quickly settles on the top positions while continuing to
experiment with different articles down the order.

To assess the quantitative dependence of the regret on the smallest
CTR \( p(b) \), we conduct an experiment varying \( p(2) \) (for \( b = 2 \)), shown
in Figure 3.5a. We note that the ratio of ideal clicks to clicks garnered
by CGPRANK increases as \( p(2) \) decreases. While our theoretical results
suggests a much stronger dependence on \( p(b) \), the effect is not as
dramatic in the experiments. This is explained by the fact that our
bounds are high probability bounds and in reality, the feedback is more
benign.

3.8 SUMMARY

In this chapter, we presented the list recommendation problem setting
and posited it as solving an exploration–exploitation dilemma. We
studied the Xbox recommender system and identified problems with
traditional recommender systems approaches using a few illustrative
insights. We then formalized the personalized list recommendation task
as a contextual multi–arm bandit problem and provided an efficient
algorithm, CGPRANK. Using Gaussian processes as a modeling tool
and kernels to capture similarity, CGPRANK efficiently navigates the
exploration–exploitation tradeoff. We proved strong performance guar-
antees for CGPRANK in terms of its overall regret and also the average
regret per slot. Our experiments support the theory presented and also
substantiate our hypothesis that parallelizing exploration across lists
can help speed up convergence and corroborate the theoretical results
we presented.
In this chapter, we study the problem of constructing a set of items choosing one item in each iteration and observing its utility. Additionally, we are given a budget on the total number of the selected items and the objective is to pick a subset of maximal value. We term the general problem setting AVID - Adaptive Valuable Item Discovery and it generalizes several important problems such as multi-arm bandits and active search. We present the problem in its simplest form in this chapter and present an algorithm, GP-Select, which requires prior knowledge about similarity between items, expressed as a kernel function. The inherent exploration (estimating the unknown value of items)–exploitation (selecting items of high value) tradeoff in AVID is balanced using Gaussian process prediction. We provide strong guarantees on the performance of GP-Select and apply it to two real-world case studies of industrial relevance: (1) Refreshing a repository of prices in a Global Distribution System for the travel industry, (2) Identifying diverse, binding-affine peptides in a vaccine design task.

4.1 INTRODUCTION

Consider a large collection of items. Each item in the collection has an inherent value that is unknown a priori. Our objective is to select a subset of items of maximal total value, subject to a constraint on the total number of the selected items, $b$. If the values of the items are known, this problem can be solved by just sorting the items in order of their values and picking the top $b$ items. But what if the values are not known? Concretely, we consider the setting where we can choose an item, observe a noisy estimate of its value, then choose and evaluate a second item and so on, until our budget is exhausted. It is clear that in order to achieve non-trivial performance, we must be able to make predictions about the value of non-selected items given observations made so far. Hence, we will assume that we are given some information about the similarity of items (e.g., via features), whereby similar items are expected to yield similar value. As
a motivating application, consider experimental design, where we may need to explore a design space, and wish to identify a set of optimal designs, evaluating one design at a time. In the early stages of medical drug development, for example, candidate compounds are subject to various tests and a fixed number of them are selected to the next stage to perform animal/human testing. Even the initial tests are expensive and the goal is to reduce the number of compounds on which these tests are conducted while still selecting a good set of compounds to promote to the next level. Another application is recommender systems, where for a given customer, we may seek to iteratively recommend items to read/watch, aiming to maximize the cumulative relevance of the entire set. Alternatively, we might want to pick users from our user base or a social network to promote a given item. In this setting, how should we select items to maximize total utility?

We will call this general class of problems AVID - Adaptive Valuable Item Discovery. To solve AVID, we need to address an exploration–exploitation dilemma, where we must select items that maximize utility (exploit) while simultaneously estimating the utility function (explore). Notice that the setting here is different from the one in Chapter 3. There we posited the recommendation problem as one of selecting a set/list of items in each round and obtained feedback for each of the items in the list at once. However, in AVID, we construct the set of items one at a time receiving feedback and adjusting our belief about the other items. We address these challenges by using ideas from Gaussian Process optimization and multi-armed bandits to provide a principled approach to AVID with strong theoretical guarantees. We handle the simplest version of this problem setting in this chapter. In Chapter 5, we introduce a few interesting and useful extensions of this setting and discuss solutions. Specifically, we introduce a novel algorithm, GP-Select, for discovering high value items in a very general setting. GP-Select can be used whenever the similarity between items can be captured by a positive definite kernel function, and the utility function has low norm in the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel. The algorithm models the utility function as a sample from a Gaussian process distribution, and uses its predictive uncertainty to navigate the exploration–exploitation tradeoff via an upper confidence based sampling approach.

In this chapter, we evaluate GP-Select in two real-world case studies. We first demonstrate how GP-Select can be used to maintain an accurate
repository of ticket prices in a Global Distribution System that serves a large number of airlines and travel agencies. Here the challenge is to selectively recompute ticket prices that likely have changed, under a budget on the number of computations allowed. Secondly, we demonstrate how GP-Select is able to determine a diverse set of candidate designs in a vaccine design application exhibiting high binding affinity to their target receptors. Our experiments highlight the efficacy of GP-Select and its applicability to a variety of problems relevant to practitioners.

**Related work**  In (budgeted) active learning, the objective is to learn a function (regression or classification) as well as possible given a limited number of queries. However, under the AVID setting, we do not seek to learn the function accurately, but only to choose items that maximize the cumulative value (e.g., the number of positive examples) of a function. That is, even if there are major differences between the actual utility and the estimated utility for many items, it does not hurt our objective as long as these are not items that would potentially be selected in the final optimal subset.

Active Search aims to discover as many members of a given class as possible under sampling budget constraints. Garnett et al. (2012) propose single and (computationally expensive) multi-step look ahead policies. It is not clear however how their approach can be applied to regression settings. Furthermore, they do not provide any performance guarantees. Xuezhi Wang et al. (2013) extended this approach to present a myopic greedy algorithm that scales to thousands of items. Warmuth et al. (2003) proposed a similar approach based on batch-mode active learning for drug discovery. The algorithms proposed in these works are similar to our exploit-only baseline and further, work only for classification tasks.

Multi-arm bandit (MAB) problems introduced in Chapter 2 abstract the explore – exploit dilemma. In contrast to the AVID setting, in MAB, arms can be selected repeatedly: Choices made do not restrict arms available in the future. In fact, our setting is a strict generalization of the bandit problem as we claim later in Section 4.2.2. We build on and extend the techniques used by Srinivas et al. (2012) in our work. In other extensions (e.g. (Kale et al. 2010; Streeter, Golovin, and Krause 2009)), the authors consider picking multiple arms per round. However, in these settings, subset selection is a repeated task with the same set of
arms available for selection each time. Also, Kleinberg, Niculescu–Mizil, et al. (2010) consider the case where only a subset of arms are available in each round. However, their results do not apply to our case where arms becomes unavailable upon being selected just once.

4.2 Problem Setup

We are given a set $V = \{v_1, \ldots, v_n\}$ of $n$ objects. There is a utility function $f : V \rightarrow \mathbb{R}_{\geq 0}$ that assigns a non-negative value to every item in the set. Given a subset $S \subseteq V$, its value $F(S) = \sum_{v \in S} f(v)$ is the sum of the values of the selected items. Given a cardinality constraint, $b > 0$, our goal is to select

$$S_b^* = \arg\max_{|S| \leq b} F(S),$$

i.e., a subset of maximum value, with cardinality bounded by $b$.

If we knew the utility function $f$, then Problem (5.6) can be easily solved by sorting the elements by their utility and picking the top-$b$. But what if we do not know $f$? In this case, we consider choosing a subset $S$ in a sequential manner. We pick one item at a time, after which the value of the selected item is revealed (possibly perturbed by noise), and can be taken into account when selecting further items. We term this sequential problem AVID - Adaptive Valuable Item Discovery.

Equivalent to maximizing the cumulative value $F(S)$, we aim to minimize the regret, i.e., the loss in cumulative value compared to an omniscient optimal algorithm that knows $f$. Formally, the regret of a subset $S_b$ of size $b$ is defined as: $\mathcal{R}_b = F(S_b^*) - F(S_b)$. We seek an algorithm whose regret grows slowly (sublinearly) with the budget $b$, so that the average regret $\mathcal{R}_b / b$ goes to 0.

4.2.1 Regularity Assumptions

In the general case, where $f$ can be any function, it is hopeless to compete against the optimal subset since, in the worst case, $f$ could be adversarial and return a value of 0 for each of the items selected by the algorithm, and positive utility only for those not selected. Hence, we make some natural assumptions on $f$ such that the problem becomes tractable. In practice, it is reasonable to assume that $f$ varies ‘smoothly’ over the candidate set $V$ such that similar items in $V$ have similar $f$
values. As we have done in Chapter 3, we assume that the similarity, 
\( k(v, v') \) of any pair of items \( v, v' \in V \) is given by a kernel function, 
\( k : V \times V \to \mathbb{R} \). We further assume that \( f \) has low “complexity” as 
measured by the norm in the Reproducing Kernel Hilbert Space (RKHS) 
associated with kernel \( k \). Recall that the RKHS \( \mathcal{H}_k(V) \) is a complete 
subspace of \( L_2(V) \) of ‘smooth’ functions with an inner product \( \langle \cdot, \cdot \rangle_k \) 
s.t \( \langle f, k(v, \cdot) \rangle = f(v) \) for all \( f \in \mathcal{H}_k(V) \). By choosing appropriate kernel 
functions, we can flexibly handle items of different types (vectors, 
strings, graphs etc.). We use the notation \( K \) to refer to the 
\( n \times n \) kernel (Gram) matrix obtained by evaluating \( k(v, v') \) for all pairs of items.

4.2.2 Explore-Exploit Tradeoff

Given the regularity assumptions about the unknown function \( f \), the 
task can be intuitively viewed as one of trading off exploration and 
exploitation. That is, we can either greedily utilize our current knowl-
gedge of \( f \) by picking the next item predicted to be of high value, or 
we can choose to pick an item that may not have the highest expected 
value but most reduces the uncertainty about \( f \) across the other items. 
This challenge is akin to the dilemma faced in multi-arm bandit prob-
lems. An important difference in our setting, motivated by practical 
considerations, is that we cannot select the same item multiple times. As 
a consequence, classical algorithms for multi-armed bandits (such as 
UCB1 of Auer, Cesa–Bianchi, and Fischer (2002) or GP-UCB of Srinivas et al. (2012)) cannot be applied, since they require that repeated 
experimentation with the same “arm” is possible. In fact, our setting 
is strictly more general than the bandit setting: We can allow repeated 
selection of a single item \( v \) by just creating multiple, identical copies 
\( v^{(1)}, v^{(2)}, \ldots \) with identical utility (i.e., \( f(v^{(1)}) = f(v^{(2)}) = \ldots \)), which 
can be modeled using a suitably chosen kernel.

Nevertheless, we build on ideas from modern bandit algorithms that 
exploit smoothness assumptions on the payoff function. In particular, 
Srinivas et al. (2012) show how the explore-exploit dilemma can be 
addressed in settings where, as in our case, the reward function has 
bounded RKHS norm for a given kernel function \( k \).

We interpret the unknown value function \( f \) as a sample from a 
Gaussian Process (GP) prior (Rasmussen and Williams 2005), with prior 
mean \( 0 \) and covariance function \( k \). Recall from Chapters 2 and 3 that 
as a consequence of this interpretation, we model the function as a
collection of normally distributed random variables, one for each item. They are jointly distributed, such that their covariances are given by the kernel:

$$\text{Cov}(f(v), f(v')) = k(v, v').$$

This joint distribution then allows us to make predictions about unobserved items via Bayesian inference in the GP model. Suppose we have already observed feedback $Y_t = \{y_1, \ldots, y_t\}$ for $t$ items $S_t = \{v_1, \ldots, v_t\}$, i.e., $y_i = f(v_i) + \epsilon_i$, where $\epsilon_i$ is independent, zero-mean Gaussian noise with variance $\hat{\sigma}^2$.

Then, for each remaining item $v$, its predictive distribution for $f(v)$ is Gaussian. Recall that the mean and variance (using noise variance $\hat{\sigma}$, according to our assumptions) can be computed by:

$$\mu_t(v) = k_t(v)^T(K_t + \hat{\sigma}^2 I)^{-1}y_t, \quad (4.2)$$

$$\sigma^2_t(v) = k(v, v) - k_t(v)^T(K_t + \hat{\sigma}^2 I)k_t(v), \quad (4.3)$$

where $k_t(v) = [k(v_1, v), \ldots, k(v_t, v)]^T$, $K_t$ is the positive semi-definite kernel matrix such that for $i, j \leq t$, $K_{t,i,j} = [k(v_i, v_j)]$ and $I$ is the $t \times t$ identity matrix. Note that while we propose a Bayesian algorithm (using a GP prior, and Gaussian likelihood), we prove agnostic results about arbitrary functions $f$ with bounded norm, and arbitrary noise bounded by $\hat{\sigma}$.

4.3 The GP-Select Algorithm

If the values are known, a greedy algorithm adding items of maximal value solves Problem (4.1) optimally. Our key idea in the unknown value case is to mimic this greedy algorithm. Instead of greedily adding the item $v$ with highest predicted gain $\mu_{t-1}(v)$, we trade exploration and exploitation by greedily optimizing an optimistic estimate of the item’s value. Concretely, our algorithm GP-Select performs both a model update and selects the next item upon receiving feedback for the current selected item. The model update is performed according to Equations (4.2) and (4.3).

For our selection rule, we use upper confidence bound sampling. Concretely, we choose

$$v_t = \arg\max_{v \in V \setminus \{v_{1:t-1}\}} \mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v), \quad (4.4)$$
Algorithm 4.1 GP-Select

Input: Ground Set $V$, kernel $k$ and budget $b$
Initialize selection set $S$

for $t = 1, 2, \ldots, b$ do
  Model Update:
  $[\mu_{t-1}(.), \sigma^2_{t-1}(.)] \leftarrow \text{GP-Inference}(k, (S, y_{\{1:t-1\}}))$

  Item Selection:
  Set $v_t \leftarrow \text{argmax}_{v \in V/\{v_1: t-1\}} \mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v)$
  $S \leftarrow S \cup \{v_t\}$
  Receive feedback $y_t = f(v_t) + \epsilon_t$

end for

The tradeoff between exploration and exploitation is implicitly handled by the time varying parameter $\beta_t$. We define the appropriate scaling factor $\beta_t$ in Theorem 4.1. Recall that $\beta_t$ alters the weighting of the posterior mean (favoring exploitation by selecting items with high expected value) and standard deviation (favoring exploration by selecting items that we are uncertain about). $\beta_t$ is chosen such that $\mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v)$ is a high-probability upper bound on $f(v)$, explained further below. Note that $\beta_t$ as defined for the original GP-UCB algorithm is computed for the setting that the arm selection is with replacement. We have to scale this appropriately if the feedback for the current arm cannot be observed again.

4.3.1 Regret bounds

We now present bounds on the regret $R_b$ incurred by GP-Select. Crucially, they do not depend on the size of the ground set $|V|$, but only on a quantity $C_K$ that depends on the task specific kernel capturing the regularity of the utility function over the set of items. Specifically, for a kernel matrix $K$, the quantity $C_K$ is given by:

$$C_K = \frac{1}{2} \log |I + \delta^{-2}K|.$$  \hspace{1cm} (4.5)

We now present the main result about GP-Select.

Theorem 4.1. Let $\delta \in (0, 1)$. Suppose that the function $f$ lies in the RKHS $H_k(V)$ corresponding to the kernel $\kappa(v, v')$ with an upper bound on
the norm of \( f \) w.r.t. \( \kappa \) given by \( M \) (i.e., \( \|f\|_{\kappa} \leq M \)). Further suppose that the noise has zero mean conditioned on the history and is bounded by \( \hat{\sigma} \) almost surely. Let \( \beta_t = 2M^2 + 300C_K \log^2(t/\delta) \). Running GP-Select with a GP prior using mean zero, covariance \( k(v, v') \) and noise model \( N(0, \hat{\sigma}^2) \), we obtain a regret bound of \( O^*(\sqrt{b(M^2 \sqrt{C_K} + C_K)}) \) w.h.p. Specifically,

\[
\Pr\{R_b \leq \sqrt{C_1 b \beta_b C_K} \quad \forall b \geq 1 \} \geq 1 - \delta
\]

where \( C_1 = \frac{8}{\log(1+\hat{\sigma}^{-2})} \).

**Proof:** Our proof builds on the analysis of Srinivas et al. (2012), who address the multi-armed bandit setting with RKHS payoff functions. A difference in our analysis is the usage of the constant \( C_K \) instead of \( \gamma_t \). According to the definition in Srinivas et al. (2012), \( \gamma_t \) measures the maximum mutual information \( I(f_S, y_S) = \frac{1}{2} \log \left| I + \sigma^{-2}K_{S,S} \right| \) that can be extracted about \( f \) using \( t \) samples \( y_S \) from \( V \).

\[
\gamma_b = \max_{S \subset V, |S| \leq b} I(f_S, y_S) \tag{4.6}
\]

But note that the way we have defined \( C_K \), it is easy to see that it is an upper bound on \( \gamma_t \). This is because, we can always define a kernel matrix within only the most informative subset of size \( t \) (say \( K' \)) and its corresponding \( C_{K'} \) and this would be exactly be \( \gamma_t \). And, \( C_K \geq C_{K'} \). This is because given the constraint of our problem setup, after \( t \) rounds, the algorithm necessarily has to have picked \( t \) distinct items to evaluate.

Apart from this, there are two important, interrelated changes from the original setting described in Srinivas et al. (2012):

1. We must respect the additional constraint that we cannot pick the same item twice.
2. The hindsight optimal choice is not a single action but instead a subset of elements in \( V \).

With these two changes, in order to prove the statement of the theorem, we need to prove a different statement of Lemma 5.2 from Srinivas et al. (2012). The remaining part of the proof (Theorem 6, Lemmas 5.3 and 5.4) remain the same as in Srinivas et al. (2012). For the sake of the proof of Theorem 4.1, we replace Lemma 5.2 from Srinivas et al. (2012) with the following Lemma 4.1 and prove a new statement that captures the main differences between the settings. Theorem 4.2, Lemmas 4.2
and 4.3 are stated without proof and correspond exactly to Theorems 6, 5.3 and 5.4 of Srinivas et al. (2012)

The first theorem establishes high probability bounds on the utility function \( f \). These carry over without modification.

**Theorem 4.2** (Srinivas et al., 2012). Let \( \delta \in (0, 1) \). Assume noise variables \( \epsilon_t \) are uniformly bounded by \( \hat{\sigma} \). Define:

\[
\beta_t = 2\|f\|^2_k + 300C_K \log^3(t/\delta)
\]

Then, for all \( v \in V \), \( b \geq 1 \)

\[
|f(v) - \mu_{t-1}(v)| \leq \beta_t^{1/2} \sigma_t(v)
\]

holds with probability \( \geq 1 - \delta \).

The next lemma bounds the instantaneous regret in terms of the widths of the confidence interval at the selected item.

**Lemma 4.1.** Fix \( t \in [1, b] \). If for all \( v \in V : |f(v) - \mu_{t-1}(v)| \leq \beta_t^{1/2} \sigma_{t-1}(v) \), then the instantaneous regret \( r_t \) is bounded by \( 2\beta_t^{1/2} \sigma_{t-1}(v_t) \).

**Proof:** At any iteration, \( t \leq b \), by the definitions of \( v_t \) and \( v_t^\ast \), one of the following statements is true.

1. Our algorithm has already picked \( v_t^\ast \) in an earlier iteration. In this case, \( \exists \ t' \ s.t. f(v_t^\ast) \geq f(v_t) \). This is because the ideal ordering has a non-increasing \( f \) value for its elements. Hence,

\[
\mu_{t-1}(v_t) + \beta_t^{1/2} \sigma_{t-1}(v_t) \geq \mu_{t-1}(v_t^\ast) + \beta_t^{1/2} \sigma_{t-1}(v_t^\ast) \\
\geq f(v_t^\ast) \\
\geq f(v_t^\ast)
\]

2. Our algorithm has not yet picked \( v_t^\ast \) in an earlier iteration. In this case,

\[
\mu_{t-1}(v_t) + \beta_t^{1/2} \sigma_{t-1}(v_t) \geq \mu_{t-1}(v_t^\ast) + \beta_t^{1/2} \sigma_{t-1}(v_t^\ast) \\
\geq f(v_t^\ast)
\]

Thus, in both cases, the statement of the lemma holds.
Lemma 4.2 (Srinivas et al., 2012). The information gain for the objects selected can be expressed in terms of the predictive variances. If $f_b = (f(v_i)) \in \mathbb{R}^b$:

$$I(y_T; f_b) = \frac{1}{2} \sum_{t=1}^{b} \log(1 + \delta^{-2} \sigma_{t-1}^2(v_i))$$

Lemma 4.3 (Srinivas et al., 2012). Pick $\delta \in (0, 1)$ and let $\beta_t$ be as defined in Theorem 4.2. Then, the following holds with probability $\geq 1 - \delta$

$$\sum_{t=1}^{b} r_t^2 \leq \beta_b C_1 I(y_b; f_b) \leq C_1 \beta_b C_K \forall b \geq 1$$

Now, using Cauchy-Schwartz inequality, $\mathcal{R}_b^2 \leq b \sum_{t=1}^{b} r_t^2$ and this proves the statement of Theorem 4.1.

4.3.2 Interpretation of the Theorem

Theorem 4.1 guarantees that under sufficiently regular $f$ and suitable choice of $\beta_t$, the average regret compared to the best subset approaches 0 as $b$ increases. Our regret bound depends only on the constant $C_K$ rather than the actual size of the set $V$. It is instructive to think of how the value $C_K$ grows as the size of the ground set, $n = |V|$ increases. As long as the kernel function is bounded, it can be seen that $C_K$ is $O(n)$. For many commonly used kernel functions, however, this quantity grows strictly sublinearly in the number $n$ of elements. For instance, for the popular RBF kernel in $d$ dimensions (that is, $V \subseteq \mathbb{R}^d$), it holds that $C_K = C_K(n) = O((\log n)^{d+1})$. Refer to Srinivas et al. (2012) for this and other analytical bounds for other kernels. In any case, a problem specific $C_K$ can always be computed efficiently using the formula in Equation (4.5). Further note that as long as we use a universal kernel $\kappa$ (like the commonly used Gaussian kernel), for finite item sets (as we consider here) the RKHS norm $||f||_\kappa$ is always bounded. Hence, Theorem 4.1 guarantees that our regret will always be bounded for such kernels, provided we choose a large enough value for $M$.

An important point to be made here is that the value of $\beta_t$ as prescribed by Theorem 4.1 is chosen very conservatively for sake of the theoretical analysis. For most practical applications, $\beta_t$ can be scaled down to achieve faster convergence and lower regret.
Figure 4.1: **a:** While average regret decreases for all non-naive algorithms, GP-Select drops much earlier and continues to outperform the baselines in the vaccine design task. **b:** GP-Select outperforms benchmarks on the fare change prediction task.

### 4.3.3 Similarities between CGPRANK and GP-Select

Note that there are strong similarities between AVID and the problem setting introduced in Chapter 3. In both cases, we use kernels to model similarity between items and Gaussian processes to model smoothness. While the objective of AVID is selecting a subset of high value by choosing one item at a time and observing feedback, in the recommendation setting, we had to select a subset/list every iteration. If one ignores the contextual aspect of CGPRANK, it is easy to visualize it as implementing GP-Select every iteration. However, we do not observe feedback as each item is picked. Hence, we simulate this feedback in an unbiased manner with the current estimated mean feedback. This problem of delayed feedback per iteration in the recommendation setting is offset by the fact that we can pick each item multiple times across iterations (but not within the same iteration). In fact, a modified one time selection constraint also makes sense in the recommendation setting if we can reliably identify users (by user ID instead of only context features). Under this modification, once an item has received a positive feedback (click/purchase), we can stop recommending it again in further iterations.
4.4  EXPERIMENTS

4.4.1  Case Study I: Airline Price Update Prediction Task

Amadeus IT group SA\(^1\) is a Global Distribution System (GDS) for airline prices. One of the services provided by Amadeus is finding the cheapest return fare between cities X and Y on requested dates of travel. This is currently done by frequently querying all the airlines for their respective cheapest fares for each pair of cities and then aggregating the results to maintain this information. This consumes a lot of bandwidth and time. Also, computing the fare for a given request is a computationally expensive task as the cheapest fare might include multiple hops possibly operated by different airlines. Hence, a table of precomputed current best prices is maintained in order to quickly respond to fare requests by customers. Since the database is typically very large and computing fares is relatively expensive in terms of computation and network bandwidth, it is challenging to frequently recompute all fares (i.e., update the entire table). Since similar prices for similar fare requests (table entries) often change at the same time, the goal is to selectively recompute only entries that changed. This task can be naturally captured in our setting, where items correspond to table entries selected for recomputation, and the utility of an item is 1, if the entry changed and 0, otherwise.

The data provided by Amadeus for this task was collected in December 2011. It consists of cheapest fares computed for 50,000 routes (origin-destination pairs) and for all departure dates up to 90 days into the future. For each departure date, the return date could be up to 15 days after the departure. The budget for selection corresponds to the total number of price refresh computations allowed. Our performance metric is the ratio between the total number of correct prices (i.e., correct entries in the table) and the total number of prices in the repository. Since we have the data with all the correct prices, we are able to compute the number of prices an algorithm would have missed to update (regret).

In our experiments, we pool all the data for a given route together, and sequentially process the data set, one “current date” at a time. The task is to discover items (table entries) that have changed between the

\(^1\) http://www.amadeus.com
current date and the next date. We thus instantiate one instance of the active discovery problem per route per day. For each instance, we select from $90 \cdot 15 = 1350$ prices to recompute. Typically only $22\%$ of the data changed between days, hence even with a budget of $0$, around $78\%$ of the prices are correct. In order to capture similarity between items (table entries), we use the following features: date, origin, destination, days until departure, duration of stay, current price. We use an RBF kernel on these features and tune the bandwidth parameter using data from four routes (origin-destination pairs). We compare GP-Select against the following baselines:

1. **Random**: Naive baseline that picks points to query uniformly at random until the budget is exhausted

2. **Epsilon-First**: A Support Vector Machine (SVM) classifier is trained on a randomly sampling part of the data. Concretely, we report the values for two different settings that perform best among other options ($5\%$ and $15\%$) of the data. The SVM is then used to predict changes, and the predicted points are updated. When higher budgets are allowed, we use a weighted version of the SVM that penalizes false negatives stronger than false positives.

Figure 4.1b presents the results of our experiments. In general, GP-Select performs better than the baselines. Note that all three non-naive algorithms reach similar maximum performance as the budget is increased close to $100\%$ of the total number of items.

### 4.4.2 Case Study II: Vaccine Design Task

The second task we consider is an experimental design problem in drug design. The goal is to discover peptide sequences that bind well to major histocompatibility complex molecules (MHC). MHC molecules act as a mediator for interaction of leukocytes (white blood cells) with other leukocytes or body cells and play an important role in the immune system. In our experiments, the goal is to choose peptide sequences for vaccine design that maximizes the binding affinity to these Type I MHC molecules (Peters et al. 2006). It is known from past experiments that similar sequences have similar binding affinity responses (Widmer et al. 2010; Jacob and Vert 2008; Krause and Ong 2011). Instead of
selecting only one optimal sequence, it is an important requirement to select multiple sequences as candidates and the actual determination of the best sequence is delayed until more thorough tests are completed further down the drug testing pipeline. Hence, while the task for this dataset can also be viewed as a classification task (binders vs non-binders), we are interested in the actual value of the binding affinity and want to pick a set of peptide sequences that have maximal affinity values.

The dataset (Peters et al. 2006) consists of peptide sequences of length $l = 9$ for the A0201 task (Widmer et al. 2010) which consists of 3089 peptide sequences along with their binding affinities (IC$_{50}$) as well features describing the peptide sequences. We normalize the binding affinities and construct a linear kernel on the peptide features. The task is then to select a subset of up to 500 sequences with maximal affinities. Since this is now inherently a regression task, we used GP regression to estimate the predictive mean of the underlying function. The following baseline algorithms were considered for comparison:

1. **Random**: Naive algorithm that picks sets of size 500 uniformly at random. We repeated this 30 times and report average total affinity values.

2. **Pure Explore**: This algorithm picks the most uncertain sequence among the remaining sequences. The GP is refitted every time an observation is made.

3. **Pure Exploit**: This algorithm always picks the next sequence as the one with the highest expected affinity as computed by GP-regression and the resulting values are used to retrain the GP. This is equivalent to the one-step lookahead policy of Garnett et al. (2012). It is not feasible to implement two or three step lookahead with this large dataset.

4. **Epsilon First**: This algorithm randomly explores for a few iterations and then once the GP is trained with the observed responses, behaves exactly like Pure Exploit. Among all the options we tried, we report results for training on the first 20% of the budget (100 sequences in this case) since this performed best. A major drawback of this algorithm is that it needs to know the budget a priori. We repeated this algorithm 30 times on the data and report the average.
The results of these experiments are presented in Figure 4.1a, which displays the average regret $\mathcal{R}_b/b$. GP-Select clearly outperforms the baselines in the regret measure. The average regret drops much faster for GP-Select and continues to remain lower than all the baseline across all the iterations.

4.5 Summary

We presented a problem setting – AVID - Adaptive Valuable Item Discovery that captures multiple settings that occur in experimental design tasks. We formalized AVID as a subset selection under bandit feedback. We then presented GP-Select, a MAB style algorithm based on Gaussian process optimization. On the theoretical front, we proved strong regret bounds on the performance of GP-Select. We then evaluated GP-Select on two tasks: (1) keeping an inventory of flight prices updated while minimizing number of queries using data from Amadeus, and (2) Discovering peptide sequences with high binding affinity to MHC molecules while reducing the number of tests carried out. The results showed that GP-Select is effective in the both these tasks and outperforms all the baselines we consider.
ADAPTIVE VALUABLE ITEM DISCOVERY - EXTENSIONS

In this chapter, we look at few interesting and useful extensions of the AVID problem setting. We argue through theory and experiments that GP-Select is able to perform well in these more challenging settings with mild and intuitive modifications. In the first part of this chapter, we demonstrate the scalability of GP-Select by experimenting with a large web scale recommender dataset. In the second part of this chapter, we study the ability of GP-Select to pick diverse subsets. Diversity in selection is an important criteria in many fields (drug design, recommender systems, web search, etc.) and we use the natural notion of submodular set functions to induce diversity in the set selected by GP-Select. Finally, we consider the more challenging setting of non-uniform costs of items which generalizes the settings presented so far. We show that the strong theoretical guarantees carry over from the special cases and demonstrate GP-Select’s efficacy on a drug design dataset with simulated costs.

5.1 SCALING TO WEB SCALE DATA

The major bottleneck in using Gaussian Processes is the computation of the posterior mean and variance. There are several works that attempt to speed up GP-based algorithms (Lawrence et al. 2002; Z. Wang et al. 2013), which we can immediately benefit from. Also, our task can be inherently parallelized by distributing the computation across multiple cores/machines and a central processor collects the top UCB scores and picks the one with the best from all the machines. The reward for the chosen item along with the item itself is communicated to the worker nodes which use the information to update the posterior mean and variances.

To obtain further improvements, we adapt the idea of lazy variance updates, originally proposed in the lazy greedy algorithm for submodular maximization (Minoux 1978) and generalized by Desautels et al. (2014) for the bandit setting, and extend it with a novel failsafe variant.
Adaptive Valuable Item Discovery - Extensions

We note that the majority of the computation time is spent on computing the posterior variance update, which requires solving a linear system for each item. The key insight is that, for a given item \( v \), \( \sigma^2_t(v) \) is monotonically decreasing in \( t \). We exploit this to recompute \( \sigma(t) \) only for those items that could influence the selection in round \( t \), via use of a priority queue. That is, in every round, we lazily pick the next item \( v_t \) based on the variance bound from the previous round and update the UCB score for that item. If \( v_t \) remains the selected item with the new score, we do not need to recompute the variances for the other items. We repeat this process until we find an item whose position at the head of the priority queue does not change after recomputation of the variance. However, note that if we have to recompute for many items in one round, it might be faster to update the variance for items due to the computational overhead associated with using a priority queue (and the benefits of parallelism). Thus, we include a failsafe condition whereby on crossing a machine and task dependent threshold on the number of lazy updates in one round, we switch to the full update. Thus, we eliminate the possibility that a large number of non-contiguous updates might be much slower than one full contiguous update for all the items.

5.1.1 Experimental Setup

The Yahoo! Webscope dataset R6A \(^1\) consists of more than 45 million user visits to the Yahoo! Today module collected over 10 days in May 2009. The log describes the interaction (view/click) of each user with one randomly chosen article out of 271 articles. It was originally used as an unbiased evaluation benchmark for bandit algorithms (Li, Chu, Langford, and Schapire 2010; Vanchinathan, Nikolic, et al. 2014). Each user \( u \) and each article \( a \) is described by a 6 dimensional feature vector. That is, \( u \in \mathbb{R}^6 \) and \( a \in \mathbb{R}^6 \). Thus, each possible interaction can be represented by a 36 dimensional feature vector (obtained from the vectorized outer product of user and item features) with a click (1) or no-click (0) as the outcome. Chu, Park, et al. (2009) present a detailed description of the dataset, features and the collection methodology.

In our experiments, we consider an application where we seek to select a subset of articles to present to a subset of users. Hence, we sequentially pick user-item pairs aiming to maximize the number of

\(^1\) http://webscope.sandbox.yahoo.com/
clicks under a constraint on the number of interactions. Here, a very natural constraint is that we do not want to repeatedly show the same item to the same user. We randomly subsample 4 million user visits from the Webscope log and treat each interaction as an item with a latent reward that can be observed only when that item is picked. As baseline, we also compute the best fixed predictor of the reward given the entire log a priori. This serves as an unrealistic benchmark to compare our algorithm and other baselines against. We also compare against the other baselines used in the vaccine design task.

For GP-Select, we use the linear kernel to model similarities between the interactions. This is just the Kronecker product ($\otimes$) of the individual linear kernels on the users and items. We simulate the selection of 100,000 interactions. The total number of clicks in the dataset (of size 4 million) is 143,664, resulting in an average clickthrough rate (CTR) of about 0.0359.

![Figure 5.1: Experiments on the news recommendation dataset.](image)

(a) Maximizing clicks on a web-scale recommendation task

(b) Performance Improvements

<table>
<thead>
<tr>
<th>Naive variance update</th>
<th>Lazy variance update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. time for one update</td>
<td>5400ms (for 4m updates)</td>
</tr>
<tr>
<td>Number of updates</td>
<td>400 Billion (Predicted)</td>
</tr>
<tr>
<td>Execution Time</td>
<td>150 hours (Predicted)</td>
</tr>
</tbody>
</table>

Figure 5.1: Experiments on the news recommendation dataset. a: GP-Select outperforms all the baselines by at least 10% while almost discovering as many clicks (8768) as the hindsight ideal (8863). b: Our failsafe approach for lazy variance updates achieves almost 40X speedup.
5.1.2 Results

Of the 100,000 selected items, the hindsight-ideal algorithm discovers 8836 items that were eventually clicked on. In comparison, GP-Select discovers 8768 items beating the other baselines by at least 10%. This corresponds to a CTR of 0.0877 which is considerably higher than the average CTR in our dataset. The next best approach is the Epsilon First approach that randomly selects items for 20% of its budget and then trains a classifier to predict the reward for the remaining items. Detailed results are presented in Figure 5.1 a.

Using this technique, we achieve a reduction factor of almost 70 in the number of updates and an overall speedup of almost 40 in terms of computational time. The results are presented in Figure 5.1 b.

5.2 Selecting Diverse Subsets

In this section, we consider a natural extension of AVID, where the goal is to obtain a diverse set of items. The motivation could be that we desire robustness, fairness etc. in the selected set. This is an important requirement in many experimental design problems where, for example for reasons of robustness, we seek to identify a collection of diverse, yet high quality designs. In our drug design example, very similar compounds might cause similar side effects in the later stages of testing. Hence, we might require a certain diversity in the selected subset while still trying to maximize total value. In our work, we address the setting where our preference for diversity is quantified by a submodular function, modeling diminishing returns incurred when picking many similar items. We prove that GP-Select provides an effective tradeoff of value and diversity, establishing bounds on its regret against an omniscient algorithm with access to the unknown objective. Our results substantially expand the class of problems that can be solved with upper confidence based sampling methods – desirable for their simplicity – in a principled manner.

5.2.1 Submodular Functions and Diversity

In several important applications, we not only seek items of high value, but also want to ensure that the selected set of items is reasonably
5.2 Selecting Diverse Subsets

diverse. One way to achieve this goal is to add to our objective another term that prefers diverse sets. Concretely, we extend the scope of AVID by considering objective functions of the form:

$$F(S) = (1 - \lambda) \sum_{v \in S} f(v) + \lambda D(S). \quad (5.1)$$

Hereby, $D(S)$ is a known measure of the diversity of the selected subset $S$. Many such diversity-encouraging objectives have been considered in the literature (c.f., (Streeter, Golovin, and Krause 2009; Lin and Bilmes 2011; Yue and Guestrin 2011; Kulesza and Taskar 2012; Kim et al. 2011)). However, the methods proposed do not directly apply to our setting of AVID. We will present an algorithm that is guaranteed to choose near-optimal sets whenever the function $D$ satisfies submodularity. We present a very brief introduction to the concept of submodular functions below.

Submodularity is a natural notion of diminishing returns, capturing the idea that adding an item helps less if more similar items were already picked (Choquet 1954). Formally, a set function $D : 2^V \rightarrow \mathbb{R}$ is submodular if for every $A \subseteq B \subseteq V$ and $v \in V \setminus B$, it holds that

$$\Delta_D(v \mid A) \geq \Delta_D(v \mid B), \quad (5.2)$$

where $\Delta_D(v \mid A) \equiv D(A \cup \{v\}) - D(A)$ is called the marginal gain of adding $v$ to set $A$. $D$ is called monotone, if, whenever $A \subseteq B$ it holds that $D(A) \leq D(B)$.

$\lambda \in [0,1]$ is a tradeoff parameter balancing the relative importance of value and diversity of the selected set. In the case where $f$ is known, maximizing $D$ requires maximizing a submodular function. This task is NP-hard, but can be solved near-optimally using a greedy algorithm (Nemhauser et al. 1978). In our work, we address the novel setting where $D$ is any known submodular function but $f$ is unknown, and needs to be estimated.

5.2.2 Algorithm and Analysis

Recall that we encode the diversity requirement into the objective function as given by Equation (5.1). Hereby $f$ is an unknown function
that operates on individual elements, while $D$ is a known set function that captures the diversity of a subset. It is natural to model diversity as a submodular function.

The rationale behind using submodular functions to model diversity is based on the intuition that adding a new element provides less benefit (marginal gain) as the set of similar items already selected increases. Many functions can be chosen to formalize this intuition. In our setting, a natural monotone submodular objective that captures the similarity as expressed via our kernel, is

$$
D(S) = \frac{1}{2} \log \left| (I + \sigma_n^{-2} K_{S,S}) \right|,
$$

where $\sigma_n \geq 0$. We use this objective in our experiments. For this choice, the marginal gain of adding an element $v$ to a set $S$ is given by:

$$
\Delta_D(v \mid S) = \frac{1}{2} \log \left( 1 + \sigma_n^{-2} \sigma^2_{v|S} \right),
$$

where $\sigma^2_{v|S}$ is the predictive variance of $f(v)$ in a GP model, where the values of elements in $S$ have already been observed up to Gaussian noise with variance $\sigma_n^2$. Conveniently, while executing GP-Select, if $\sigma_n = \sigma_n$, we already compute $\sigma^2_{v|S}$ in order to evaluate the decision rule (4.4). Hence, at almost no additional cost we can compute the marginal gain in diversity for any candidate item $v$.

In order to select items that provide value and diversity, it is natural to modify the selection rule of GP-Select in the following way:

$$
v_t = \arg\max_{v \in V \setminus \{v_{1:t-1}\}} \left( (1 - \lambda) \left[ \mu_{t-1}(v) + \beta_t^{1/2} \sigma_{l-1}(v) \right] \right. \\
\left. + \lambda \Delta_D(v \mid \{v_1, \ldots, v_{t-1}\}) \right).
$$

This decision rule greedily selects item $v$ that maximizes a high-probability upper bound on the marginal gain $\Delta_F(v \mid \{v_1, \ldots, v_{t-1}\})$ of the unknown combined objective $F$.

**Regret Bound** The regret bound for GP-Select in Chapter 4 depended on the fact that we were optimizing against $f$ that assigned values to individual elements, $v \in V$. The same bounds need not hold in the more challenging setting when trading value against diversity. In fact, even if both $f$ and $D$ are completely known for all $v \in V$, it turns
out that optimizing $F$ in (5.1) is NP-hard for many monotone submodular functions $D$ (Feige 1998). While finding the optimal set is hard, Nemhauser et al. (1978) states that – for a known monotone submodular function – a simple greedy algorithm provides a near-optimal solution.

Formally, suppose $S_0' = \emptyset$ and $S_i'$, the greedy extension to $S_i'$. That is, $S_{i+1}' = S_i' \cup \{\arg\max_{v \in V \setminus S_i'} \Delta F(v | S_i')\}$. Thus, $S_b'$ is the set we obtain when selecting $b$ items, always greedily maximizing the marginal gain over the items picked so far. Then it holds that $F(S_b') \geq (1 - 1/e) \max_{|S| \leq b} F(S) = (1 - 1/e) F(S_b^*)$. Moreover, without further assumptions about $D(S)$ and $f$, no efficient algorithm will produce better solutions in general. Since we are interested in computationally efficient algorithms, we measure the regret of a solution $S_b$ by comparing $F(S_b)$ to $F(S_b')$, which is the bound satisfied by the greedy solution. Formally, $R_b = (1 - 1/e) F(S_b^*) - F(S_b)$.

**Theorem 5.1.** Under the same assumptions and conditions of Theorem 4.1 from Chapter 4,

$$Pr\{R_b \leq \sqrt{C_1 b \beta b C_K} \ \forall b \geq 1\} \geq 1 - \delta,$$
where \( R_b = (1 - 1/e)F(S_b^*) - F(S_b) \) is the regret with respect to the value guaranteed when optimizing greedily given full knowledge of \( f \) and \( D \).

**Proof:** We use the proof techniques of Nemhauser et al. (1978) and its extension (Krause, Singh, et al. 2008).

Denote by \( S_i = \{v_1, \ldots v_i\} \) the solution set of GP-Select after \( i \) iterations and by \( S_i^* = \{v_1^*, \ldots v_i^*\} \), the solution set of the exact optimal solution after \( i \) iterations.

Given that \( F(S) = (1 - \lambda) \sum_{v \in S} f(v) + \lambda D(S) \), the marginal gain of GP-Select in the \((i + 1)\)th step is given by:

\[
\Delta_i = F(S_i \cup \{v_{i+1}\}) - F(S_i).
\]

Now, from Lemma 4.1 and submodularity, in each iteration, \( \Delta_i \) can differ from the best greedy choice by at most the width of the confidence interval

\[
\Delta_i \geq \max_{v \in V \setminus \{v_1 \ldots v_i\}} \left\{ F(S_{i-1} \cup \{v\}) - F(S_{i-1}) - (1 - \lambda)w_i(v_i) \right\}
\]

where \( w_i(v_i) = 2\beta_i^{1/2}\sigma_i(v_i) \).

Since \( F \) is monotone,

\[
F(S_i \cup S_b^*) \geq F(S_b^*)
\]

But also, by definition of \( S_b^* \), for all \( i = 0, \ldots, b \),

\[
F(S_i \cup S_b^*) \leq F(S_i) + b(\Delta_{i+1} + \epsilon_i) = \sum_{j=1}^{i} \Delta_j + b(\Delta_{i+1} + \epsilon_i)
\]

We can then get the following inequalities,

\[
F(S_b^*) \leq b(\Delta_1 + \epsilon_0)
\]

\[
F(S_b^*) \leq \Delta_1 + b(\Delta_2 + \epsilon_1)
\]

\[
\vdots
\]

\[
F(S_b^*) \leq \sum_{j=1}^{b-1} \Delta_j + b(\Delta_b + \epsilon_{b-1})
\]
5.2 SELECTING DIVERSE SUBSETS

Multiplying both sides of the $i^{th}$ inequality by $(1 - \frac{1}{b})^{b-1}$, and adding all the inequalities, we get

$$\left(\sum_{i=0}^{b-1} (1 - 1/b)^i\right) F(S_b^*) \leq b \sum_{i=1}^{b} (\Delta_i + \epsilon_{i-1}) = b \left(F(S_b) - \sum_{i=0}^{b-1} \epsilon_i\right)$$

Further, we can simplify this to,

$$F(S_b) - R_b \geq \left(1 - (1 - 1/b)^b\right) F(S_b^*) \geq (1 - 1/e) F(S_b^*)$$

From Theorem 4.1, we can bound $R_b = \sum_{i=0}^{b-1} \epsilon_i$ by $\sqrt{C_1 b \beta \beta} \forall b \geq 1$, thus proving the claim of Theorem 5.1.

The proof rests on interpreting GP-Select as implementing an approximate version of the greedy algorithm maximizing $\Delta_F(v \mid S_i)$. In fact, Theorem 5.1 can be generalized to a large number of settings where the greedy algorithm is known to provide near-optimal solutions for constrained submodular maximization. For example, instead of cardinality constraints, one can use GP-Select to maximize a known submodular function $D$ plus an unknown modular function $f$ subject to matroid constraints or $p$-independence systems (optimizing over spanning trees, matchings or other combinatorial structures).

As an illustration of the application of this modified GP-Select to diverse subset selection, refer to Figure 5.2. When $\lambda = 0$, GP-Select reverts back to Algorithm 4.1 and hence, picks locations only based on its expected $f$ value. This is clear from the thick bands of points sampled near the maximum. At $\lambda = 0.6$, GP-Select balances between expected $f$ values of the points and the marginal gain in diversity of the points picked $\Delta_D(v \mid S)$. At $\lambda$ close to 1, GP-Select picks mostly by marginal gain which will be approximately uniform if the kernel used is isotropic (e.g. Gaussian kernel).

5.2.3 Experimental Results

Using the same vaccine design dataset that we described in Chapter 4, we implement the modified version of GP-Select to select a diverse set of peptide sequences. This requirement of diversity is quite natural...
Figure 5.3: a: Performance of GP-Select in selecting diverse subsets. For different values of λ, the average regret against the greedy approximate algorithm decreases. b: Improvements in diversity can be obtained at little loss of utility.

for our drug testing application: Very similar sequences, while having similar affinity values, might also suffer from similar shortcomings in later stages of drug testing. We run GP-Select with different values of the tradeoff parameter λ, and report the results. Figure 5.3 a, is the average regret $R_b/b$ of GP-Select for different values of $\lambda$. The plot demonstrates that when selecting diverse subsets GP-Select has a similar regret performance as in the initial case when it was selecting only for value. Also, the average regret compared to the greedy optimal solution slightly increases with increase in the value of $\lambda$. Figure 5.3 b shows the inherent tradeoff between value and diversity. We use values of $\lambda = \{0, 0.5, 0.75, 0.875, 0.9375, 0.96875\}$ and plot the performance. We use the diversity function defined in Equation (5.1). It should be noted that this function is in log scale. From the plot it is clear that for a significant increase in the diversity score, we lose very little functional value, which suggests that robustness of the solution set can be achieved at very little cost. The greedy curve on this same plot shows the tradeoff that the greedy algorithm obtains knowing the utility function. This result serves as a reference, as no efficient algorithm can match it without actually knowing the response function over all the sequences. Note that as we put all weight on diversity, as expected, GP-Select’s performance converges to that of the greedy algorithm.
5.3 NON-UNIFORM COSTS

In all the settings of AVID so far, all items had a uniform cost of being selected. However, there are many settings in which this might not be the case. For instance in experimental design, testing different items might have different costs. In drug design, the candidate sequences might have different tests to ascertain their suitability and/or affinity for the task. We believe this is a novel problem that generalizes the basic AVID setting introduced in Chapter 4. Note that this is different from the budget limited explore-exploit problems that have been studied in context of the stochastic knapsack problem where the learning process is constrained by available resources. Gupta et al. (2011) provide strong regret bounds for the scalar budget case in this setting. Tran–Thanh et al. (2010) consider prior-free learning for the same problem. Badanidiyuru, Kleinberg, et al. (2013) study the problem under multi-dimensional budget constraints. However, all these approaches consider arms as independent (i.e., uncorrelated), and hence do not generalize observations across similar arms as we do. Specifically, they do not work in the setting where each item can only be chosen once as there is no mechanism to generalize the feedback across the other items.

In this section, we consider the case where there is a function \( c : V \rightarrow \mathbb{R}_{>0} \), assigning a positive cost \( c_v = c(v) \in [c_{\min}, c_{\max}] \) to each item \( v \in V \). We are also subject to a budget constraint \( b \) which represents the maximum cumulative cost of the selected subset. We represent by \( C(S) \) the cumulative costs of the items. Thus, our objective is to select:

\[
S_b^* = \arg\max_{C(S) \leq b} F(S),
\]  

i.e., a subset of maximum value, with cost bounded by \( b \). If we knew the utility function \( f \), then Problem (5.6) is the classical knapsack problem. While NP-hard, for any \( \varepsilon \), an \( \varepsilon \)-optimal solution can be found via dynamic programming. Since we do not know \( f \), we consider choosing \( S \) in a sequential manner and extend GP-Select in an intuitive manner to handle non-uniform costs.

5.3.1 Algorithm and Analysis

In the general case where each element \( v \in V \) has different costs of selection \( c_v \), the budget \( b \) is the maximum allowed total cost of all items.
in the selected subset. We modify the selection rule of the Algorithm 4.1 from Chapter 4 to take the estimated cost-benefit ratio into account. This selection rule is inspired by the greedy solution to the 0/1 knapsack problem. Most of the other steps remain the same except ensuring that we respect the budget, and the formula for computing $\beta_t$. The new selection rule for the setting without diversity is:

$$v_t = \arg\max_{v \in V \setminus S, c_v \leq b - C(S)} \frac{\mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v)}{c_v}.$$  \hfill (5.7)

Hence, instead of maximizing an optimistic estimate of the item’s value, we greedily maximize an optimistic estimate of the benefit-cost ratio. Note that this greedy rule encourages some natural opportunistic exploration: Initially, it will select items that we are very uncertain about (large $\sigma_{t-1}$), but that also have little cost. Later on, as the utility is more accurately estimated, it will also invest in more expensive items, as long as their expected value ($\mu_{t-1}$) is high.

The idea above can be generalized to encourage diversity as well. The selection rule in (5.5) from Section 5.2 can be modified to maximize the ratio

$$\frac{(1 - \lambda) \left[ \mu_{S-1}(v) + \beta_S^{1/2} \sigma_{S-1}(v) \right] + \lambda \Delta_D(v \mid S)}{c_v}.$$  \hfill (5.8)

Hence, in this most general setting, we greedily optimize a high-probability upper bound on the cost-benefit ratio of the marginal gain for the joint objective.

Upon these modifications, we can obtain the result presented in Theorem 5.2. The result holds for running GP-Select for selecting diverse items with items in the ground set having non-uniform costs of selection.

**Theorem 5.2.** Under the same assumptions and conditions of Theorem 4.1, running GP-Select with non-uniform costs for the items, we have that

$$\Pr\{\mathcal{R}_b \leq \left( \max_{v \in V} f(v) + c_{\max} \sqrt{C_1 b \beta_b C_K} \right) \forall b \geq 1 \} \geq 1 - \delta,$$

where $\mathcal{R}_b = (1 - 1/e) F(S^*_b) - F(S_b)$ is the regret with respect to the value guaranteed when optimizing greedily given full knowledge of $f$ and $D$. 

82
5.3 Non-uniform Costs

Proof: For ease of presentation we use $c_j$ to denote $c_{v_j}$ when there is no confusion. Also, without loss of generality, we assume that $c_{\text{min}} \geq 1$.

Our proof is adapted from Streeter and Golovin (2008). We consider a modified version of greedy algorithm that is allowed to pick from only those elements whose individual costs are less than the budget $b$. Let $(S_j)_j$ be the sequence of subsets chosen by this greedy algorithm.

$S_1 \subset S_2 \subset S_3 \ldots$. Let $l$ be the maximum index such that $C(S_l) \leq b$. We will show that $F(S_{l+1})$ is nearly optimal. And then, it is easy to see that $F(S_l) \geq F(S_{l+1}) - \max_{v \in V} f(v)$. In order to prove the theorem, we require the following lemma.

**Lemma 5.1.** If $F$ is submodular, $S^* \in V$ is the optimal subset under budget $b$, and we run the modified greedy procedure picking elements $\{v_1, v_2, \ldots\}$ in that order resulting in sets $S_1 \subset S_2 \subset S_3 \ldots$. Then,

$$F(S^*) \leq F(S_j) + bs_{j+1} + \frac{B}{c_{j+1}} \epsilon_{j+1}$$

where $s_{j+1} = \frac{F(S_{j+1}) - F(S_j)}{c_{j+1}}$ and $\epsilon_{j+1}$ is the error in estimating $s_j$.

Proof: Let $S^* \setminus S_j = \{o_1, o_2, \ldots o_m\}$

Then,

$$F(S^*) \leq F(S_j \cup S^*)$$

$$\leq F(S_j) + \sum_{i=1}^{m} \Delta(o_i | S_j)$$

$$\leq F(S_j) + b \left[ \frac{F(S_{j+1}) - F(S_j) + \epsilon_{j+1}}{c_{j+1}} \right]$$

$$= F(S_j) + bs_{j+1} + \frac{b}{c_{j+1}} \epsilon_{j+1}$$

where the second inequality is due to submodularity and the third inequality is due to the greedy selection rule.

For running GP-Select, $\epsilon_{j+1}$ is instantaneous regret which is upper bounded by the width of the confidence interval, $2\beta_{S_j}^{1/2} \sigma_{S_{j-1}}(v_j)$

Now we are ready to prove Theorem 5.2
Consider $S_{l+1}$, the result of greedy algorithm that just becomes infeasible. Let $\Delta_j = F(S^*) - F(S_j)$

$$\Delta_j \leq b s_{j+1} + \frac{b}{e_{j+1}} \epsilon_{j+1} \quad \text{(From Lemma 5.1)}$$

$$= b \left( \frac{\Delta_j - \Delta_{j+1}}{e_{j+1}} + \epsilon_{j+1} \right)$$

Rearranging the terms, we get, $\Delta_{j+1} \leq \Delta_j \left( 1 - \frac{c_{j+1}}{b} \right) + c_{j+1} \epsilon_{j+1}$

Using the fact that $1 - \frac{c_{j+1}}{b} \leq 1$, and using the telescopic sum, we get,

$$\Delta_{l+1} \leq \Delta_1 \left( \prod_{j=1}^{l} 1 - \frac{c_{j+1}}{b} \right) + \sum_{j=1}^{l} (c_{j+1} \epsilon_{j+1})$$

Note that the product series is maximised when $c_{j+1} = \frac{b}{l}$. Thus,

$$\Delta_{l+1} \leq \Delta_1 \left( 1 - \frac{1}{l} \right)^l + \sum_{j=1}^{l} (c_{j+1} \epsilon_{j+1})$$

$$\leq \Delta_1 \frac{1}{e} + \sum_{j=1}^{l} (c_{j+1} \epsilon_{j+1})$$

$$\leq F(S^*) \frac{1}{e} + \sum_{j=1}^{l} (c_{j+1} \epsilon_{j+1})$$

$$\leq F(S^*) \frac{1}{e} + c_{\text{max}} \sum_{j=1}^{l} \epsilon_{j+1}$$

$$\leq F(S^*) \frac{1}{e} + c_{\text{max}} R_b$$

Thus, $F(S_{l+1}) > (1 - \frac{1}{e}) F(S^*) - c_{\text{max}} R_b$ and $F(S_l) \geq F(S_{l+1}) - \max_{v \in V} f(v)$

5.3.2 Experimental Results

The vaccine design task that was described in detail in Chapter 4 also provides a natural motivation for the non-uniform costs setting. Typically, the cost of testing depends on the actual sequence being tested. Also, field tests differ markedly in their cost of execution. For
5.4 summary

This chapter builds on Chapter 4 and extends the applicability of GP-Select. We first presented techniques and results for scaling GP-Select to large datasets. Using a mix of computational and statistical techniques, we showed how GP-Select can be used to select positive interactions in a recommender systems task. Crucially, we obtained almost 40X speedup in the running time. We then extended the AVID setting to include a preference for diversity in the selected subset. We use concepts from submodularity to model and optimize for this preference. Our results show that for little loss in the cumulative value, one can
obtain diverse subsets by tuning the linearization parameter $\lambda$. Finally, we generalized AVID to the setting where testing costs are non-uniform and item–specific. Under this setting a modified version of GP-Select still retains favourable regret performance and we empirically verified our claims through simulated experiments.
As presented in Chapter 2, partial monitoring is a general model for online learning with limited feedback. In each round, an agent chooses an action and the opponent chooses an outcome. At the end of each round, the agent suffers some loss and receives some feedback based on the action and the outcome. The goal of the agent is to minimize her cumulative loss. Applications range from dynamic pricing to label-efficient prediction to dueling bandits. In this chapter, we assume that we are given some prior information about the distribution based on which the opponent generates the outcomes. We propose BPM, a family of new efficient algorithms whose core is to track the outcome distribution with an ellipsoid centered around the estimated distribution. We show that our algorithm provably enjoys near-optimal regret rate for locally observable partial-monitoring problems against stochastic opponents. As demonstrated with experiments on synthetic as well as real-world data, the algorithm outperforms previous approaches, even for very uninformed priors, with an order of magnitude smaller regret and lower running time.

6.1 INTRODUCTION

We consider Partial Monitoring, a repeated game where in every time step an agent chooses an action while, simultaneously, an opponent chooses an outcome. Then the player receives a loss based on the action and outcome chosen. The learner also receives some feedback based on which she can make better decisions in subsequent time steps. The goal of the learner is to minimize her cumulative loss over some time horizon.

The performance of the learner is measured by the regret, the excess cumulative loss of the learner compared to that of the best fixed constant action. As with previous problems, our goal is to achieve Hanan consistency for the agent’s strategy.

Partial monitoring generalizes most problems in online learning with full information and also encompasses the MAB problem. However,
there are many problems in online learning that do not fall into either of these feedback structures. An important example for a problem that does not fit in either full-information or bandit problems is *dynamic pricing*. Consider the problem of a vendor wanting to sell his products to customers for the best possible price. When a customer comes in, she (secretly) decides on a maximum price she is willing to buy his product for, while the vendor has to set a price without knowing the customer’s preferences. The loss of the vendor is some preset constant if the customer did not buy the product, and an “opportunity loss”, when the product was sold cheaper than the customer’s maximum. The feedback, on the other hand, is merely an indicator whether the transaction happened or not.

Dynamic pricing is just one of the practical applications of partial monitoring. *Label efficient prediction* (introduced briefly in Chapter 2), in its simplest form, has three actions: the first two actions are guesses of a binary outcome but provide no information, while the third action provides information about the outcome for some unit loss as the price. This can be thought of an abstract form of *spam filtering*: the first two actions correspond to putting an email to the inbox and the spam folder, the third action corresponds to asking the user if the email is spam or not. Another problem that can be cast as partial monitoring is that of *dueling bandits* (Yue, Broder, et al. 2012; Ailon et al. 2014) in which the learner chooses a pair of actions in every time step, the loss she suffers is the average loss of the two actions, and the feedback is which action was “better”.

While most prior work in this area dealt with adversarial feedback setting, we focus in this chapter on strategies against stochastic opponents instead.

**Related work** Partial monitoring was first addressed in the seminal paper of Piccolboni and Schindelhauer (2001), who designed and analyzed the algorithm FeedExp3. The algorithm’s main idea is to maintain an unbiased estimate for the loss of each action in every time step, and then use these estimates to run the full-information algorithm (EWA). Piccolboni and Schindelhauer (2001) proved an $O(T^{3/4})$ upper bound on the regret (not taking into account the number of actions) for games for which learning is at all possible. This bound was later improved by Cesa–Bianchi et al. (2006) to $O(T^{2/3})$, who also constructed an example of a problem for which this bound is optimal.
From the above bounds it can be seen that not all partial-monitoring problems have the same level of difficulty: while bandit problems enjoy an $O(\sqrt{T})$ regret rate, some partial-monitoring problems have $\Omega(T^{2/3})$ regret. To this end, Bartók, Pál, et al. (2011) showed that partial-monitoring problems with finitely many actions and outcomes can be classified into four groups: trivial with zero regret, easy with $\tilde{\Theta}(\sqrt{T})$ regret, hard with $\Theta(T^{2/3})$ regret, and hopeless with linear regret. The distinguishing feature between easy and hard problems is the local observability condition, an algebraic condition on the feedback structure that can be efficiently verified for any problem. Bartók, Pál, et al. (2011) showed the above classification against stochastic opponents with the help of algorithm Balaton. This algorithm keeps track of estimates of the loss difference of “neighboring” action pairs and eliminates actions that are highly likely to be suboptimal.

Since then, several algorithms have been proposed that achieve the $\tilde{O}(\sqrt{T})$ regret bound for easy games (Bartók, Zolghadr, et al. 2012; Bartók 2013). All these algorithms rely on the core idea of estimating the expected loss difference between pairs of actions.

### 6.2 Problem Setup

Partial monitoring is a repeated game where in every round, a learner chooses an action while the opponent chooses an outcome from some finite action and outcome sets. Then, the agent observes a feedback signal (from some given set of symbols) and suffers some loss, both of which are deterministic functions of the action and outcome chosen. In our work we assume that the opponent chooses the outcomes in an iid stochastic manner. The goal of the agent is to minimize her cumulative loss.

The following definitions and concepts are mostly taken from Bartók, Pál, et al. (2011). An instance of partial monitoring is defined by the loss matrix $L \in \mathbb{R}^{N \times M}$ and the feedback table $H \in \Sigma^{N \times M}$, where $N$ and $M$ are the cardinality of the action set, $V = \{v_1, \ldots, v_N\}$ and the outcome set $O = \{o_1, \ldots, o_M\}$, respectively. Where there is no confusion we refer to action $v_i$ as action $i$ and outcome $o_j$ as outcome $j$. $\Sigma$ is some alphabet of symbols. That is, if learner chooses action $i$ while the outcome is $j$, the loss suffered by the learner is $L[i, j]$, and the feedback received is $H[i, j]$.

For an action $v_i \in V$, let $\ell_i$ denote the column vector given by the $i^{th}$ row of $L$. Let $\Delta_M$ denote the $M$-dimensional probability simplex. It is
easy to see that for any \( p \in \Delta_M \), if we assume that the opponent uses \( p \) to draw the outcomes (that is, \( p \) is the opponent strategy), the expected loss of action \( i \) can be expressed as \( \ell_i^\top p \).

We measure the performance of an algorithm with its \textit{expected regret}, defined as the expected difference of the cumulative loss of the algorithm and that of the best fixed action in hindsight:

\[
R_T = \max_{v_i \in V} \sum_{t=1}^T (\ell_{I_t} - \ell_{i})^\top p ,
\]

where \( T \) is some time horizon, \( I_t (t = 1, \ldots, T) \) is the action chosen in time step \( t \), and \( p \) is the outcome distribution the opponent uses.

We also assume we have some prior knowledge about the outcome distribution in the form of a \textit{confidence ellipsoid}. That is, we assume that we are given a distribution \( p_0 \in \Delta_M \) and a symmetric positive semidefinite covariance matrix \( \Sigma_0 \in \mathbb{R}^{M \times M} \) such that the true outcome distribution \( p^* \) satisfies

\[
\|p_0 - p^*\|_{\Sigma_0^{-1}} = \sqrt{(p_0 - p^*)^\top \Sigma_0^{-1} (p_0 - p^*)} \leq 1.
\]

We use the term “confidence ellipsoid” even though our condition is not probabilistic; we do not assume that \( p^* \) is drawn from a Gaussian distribution before the game starts. On the other hand, the way we track \( p^* \) is derived by Bayes updates with a Gaussian conjugate prior, hence the name. We would also like to note that having the above prior knowledge is without loss of generality. For “large enough” \( \Sigma_0 \), the whole probability simplex is contained in the confidence ellipsoid and thus partial monitoring without any prior information reduces to our setting.

The following definition reveals how we use the loss matrix to recover the structure of a game.

\textbf{Definition 6.1} (Cell decomposition, Bartók, Pál, et al. \textit{(2011, Definition 2)}). For any action \( i \in V \), let \( C_i \) denote the set of opponent strategies for which action \( v_i \) is optimal:

\[
C_i = \left\{ p \in \Delta_M : \forall v_i \in V, (\ell_i - \ell_j)^\top p \leq 0 \right\}.
\]

We call the set \( C_i \) the \textit{optimality cell} of action \( v_i \). Furthermore, we call the set of optimality cells \( \{C_1, \ldots, C_N\} \) the \textit{cell decomposition} of the game.
6.2 PROBLEM SETUP

Every cell $C_i$ is a convex closed polytope, as it is defined by a linear inequality system. Normally, a cell has dimension $M - 1$, which is the same as the dimensionality of the probability simplex. In this setting, there could be two kinds of degeneracies. Firstly a cell is of lower dimensionality than $M - 1$. Another possible degeneracy is when two actions share the same cell. In our work, for ease of presentation, we assume that these degeneracies do not appear. For an illustration of cell decomposition, see Figure 6.1a.

Now that we know the regions of optimality, we can define when two actions are neighbors. Intuitively, two actions are neighbors if their optimality cells are neighbors in the strong sense that they not only meet in “one corner”.

**Definition 6.2** (Neighbors, Bartók, Pál, et al. (2011, page 4)). Two actions $v_i$ and $v_j$ are neighbors, if the intersection of their optimality cells $C_i \cap C_j$ is an $M - 2$-dimensional convex polytope.

To optimize performance, the agent’s primary goal is to find out which cell the opponent strategy lies in. Then, the learner can choose the action associated with that cell to play optimally. Since the feedback the algorithm receives is limited, this task of finding the optimal cell may be challenging.

The next definition enables us to utilize the feedback table $H$.

**Definition 6.3** (Signal matrix, Bartók, Pál, et al. (2011, Definition 1)).

Let $\{\alpha_1, \alpha_2, \ldots, \alpha_{\sigma_i}\} \subseteq \Sigma$ be the set of symbols appearing in row $i$ of the feedback table $H$. We define the signal matrix $Z_i \in \{0, 1\}^{\sigma_i \times M}$ of action $i$ as

$$Z_i[k, j] = \mathbb{1}(H[i, j] = \alpha_k).$$

That is, $Z_i$ is the indicator table of observing symbols $\alpha_1, \ldots, \alpha_{\sigma_i}$ under outcomes $1, \ldots, M$ given that the action chosen is $i$. For an example, consider the case when the $i$th row of $H$ is $(a \ b \ a \ c)$. Then,

$$Z_i = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A very useful property of the signal matrix is that if we represent outcomes with $M$-dimensional unit vectors, then $Z_i$ can be used as a
Figure 6.1: (a) An example for a cell decomposition with $M = 3$ outcomes. Under the true outcome distribution $p^*$, action 3 is optimal. Cells $C_1$ and $C_3$ are neighbors, but $C_2$ and $C_5$ are not. (b) The current estimate $p_{t-1}$ is far away from the true distribution, the confidence ellipsoid is large. (c) After updating, $p_t$ is closer to the truth, the confidence ellipsoid shrinks.

The following condition condition is key in distinguishing easy and hard games:

**Definition 6.4** (Local observability, Bartók, Pál, et al. (2011, Definition 3)). Let actions $v_i$ and $v_j$ be neighbouring actions. These actions are said to be *locally observable* if $\ell_i - \ell_j \in \text{Im } Z_i^T \oplus \text{Im } Z_j^T$. Furthermore, a game is *locally observable* if all of its neighboring action pairs are locally observable.

Bartók, Pál, et al. (2011) showed that finite stochastic partial-monitoring problems that admit local observability have $\tilde{\Theta}(\sqrt{T})$ minimax expected regret. In our work, we present a new algorithm family that achieves the same regret rate for locally observable games against stochastic opponents.

### 6.3 BPM Family of Algorithms

As with our other algorithms, CGPRank and GP-Select, BPM also has two components. The *model update* component keeps track of a belief about the true outcome distribution and provides us with a set of *feasible* actions in every round. The *selection* component is responsible for
Algorithm 6.1 BPM

**input:** $L, H, p_0, \Sigma_0$

**initialization:** Calculate signal matrices $Z_i$

for $t = 1$ to $T$ do

- Use selection rule (cf., Section 6.3.2) to choose an action $I_t$
- Observe feedback $Y_t$
- Update posterior: $\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + P_{I_t}$ and $p_t = \Sigma_t \left( \Sigma_{t-1}^{-1} p_{t-1} + Z_{I_t}^T (Z_{I_t} Z_{I_t}^T)^{-1} Y_t \right)$

end for

for selecting the action to play from this action set. Pseudocode for the algorithm family is shown in Algorithm 6.1.

### 6.3.1 Update Rule

The actual outcomes could be generated from arbitrary distributions (with the only restriction being that they are drawn stochastic i.i.d). However, for reasons of tractability and ease of analysis, we model the outcomes as being generated from a Gaussian distribution with covariance $\Sigma = I_M$ and unknown mean. We also posit a Gaussian prior for tracking the mean. The parameters of this prior are denoted by $p_0$ (mean) and $\Sigma_0$ (covariance). In every time step, we perform a Gaussian Bayes-update using the observation received.

**full-information case** If we had complete information about the outcome in each round, the update step is identical with the standard Gaussian one-step update:

$$\Sigma_t = \Sigma_{t-1} - \Sigma_{t-1} (\Sigma_{t-1} + I)^{-1} \Sigma_{t-1}$$

or equiv.

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + I,$$

$$\mu_t = \Sigma_t \left( \Sigma_{t-1}^{-1} \mu_{t-1} + X_t \right)$$

or equiv.

$$\mu_t = \mu_{t-1} + \Sigma_t (X_t - \mu_{t-1}).$$

Here we use subindex $t-1$ for the prior parameters and $t$ for the posterior parameters in time step $t$, and denote by $X_t$ the outcome (observed in this case), encoded by an $M$-dimensional unit vector.

**general case** However, we do not have access to the actual outcome in each round. Hence, we face the problem of not observing the outcome but only some symbol that is governed by the signal matrix
of the action we chose and the outcome itself. If we denote, as above, the outcome at time step $t$ by an $M$-dimensional unit vector $X_t$, then the observation symbol can be thought of as a unit vector given by $Y_t = Z_i X_t$, provided the chosen action is $i$. It follows that what we observe is a linear transformation of the sample from the outcome distribution.

Following the Bayes update rule and assuming we chose action $i$ at time step $t$, we derive the corresponding Gaussian posterior given that the likelihood of the observation is $\pi(Y|p) \sim N(Z_i p, Z_i Z_i^T)$. One can easily see that the posterior distribution is also Gaussian with covariance $\Sigma_t = (\Sigma_{t-1}^{-1} + P_i)^{-1}$ and mean $p_t = \Sigma_t \left( \Sigma_{t-1}^{-1} p_{t-1} + P_i X_t \right)$, where $P_i = Z_i^T (Z_i Z_i^T)^{-1} Z_i$ is the orthogonal projection to the image space of $Z_i^T$. Note that even though $X_t$ is not observed, the update can be performed, since $P_i X_t = Z_i^T (Z_i Z_i^T)^{-1} Z_i X_t = Z_i^T (Z_i Z_i^T)^{-1} Y_t$.

Earlier work on partial monitoring kept track of loss differences for individual actions. However, because of the way we track the outcome distribution, we can generalize feedback from one action such that it provides information about losses across all the actions. We believe that this property has a major role in the empirical performance improvement over existing methods.

An important part in analyzing our algorithm is to show that, despite the fact that the outcome distribution is not Gaussian, the update tracks the true outcome distribution well. For an illustration of tracking the true outcome distribution with the above update, see Figures 6.1b and 6.1c.

### 6.3.2 Selection rules

For selecting actions given the posterior parameters, we propose two versions for the selection rule:

1. Draw a random sample $p$ from the distribution $N(p_{t-1}, \Sigma_{t-1})$, project the sample to the probability simplex, then choose the action that minimizes the loss for outcome distribution $p$. This rule is a close relative of Thompson-sampling. We call this version of the algorithm BPM-TS.

2. Use $p_{t-1}$ and $\Sigma_{t-1}$ to build a confidence ellipsoid for $p^*$, enumerate all actions whose cells intersect with this ellipsoid, then
choose the action that was chosen the fewest times so far (called BPM-Least).

In our experiments we demonstrate the performance of both versions. However, for our analysis we only consider version BPM-Least and leave the analysis of BPM-TS as an open problem for within the context of this thesis.

6.4 Analysis

We analyze BPM-Least that uses the Gaussian updates, and considers a set of feasible actions based on the criterion that an action is feasible if its optimality cell intersects with the ellipsoid

\[
\left\{ p : \| p - p_t \|_{\Sigma^{-1}} \leq 1 + \sqrt{\frac{1}{2} N \log MT} \right\}.
\]

From these feasible actions, it picks the one that has been chosen the fewest times up to time step \( t \). For this version of the algorithm, the following regret bound holds.

**Theorem 6.5.** Given a locally observable partial-monitoring problem \((L, H)\) with prior information \( p_0, \Sigma_0 \), the algorithm BPM-Least achieves expected regret

\[
\mathcal{R}_T \leq C \sqrt{TN \log(MT)},
\]

where \( C \) is some problem-dependent constant.

The above constant \( C \) depends on two main factors, both of them related to the feedback structure. The first one is the sum of the smallest eigenvalues of \( Z_iZ_i^T \) for every action \( i \). The second is related to the local observability condition. As the condition says, for every neighboring action pairs \( i \) and \( j \), \( \ell_i - \ell_j \in \text{Im} Z_i^T \oplus \text{Im} Z_j^T \). This means that there exist \( u_{ij} \) and \( u_{ji} \) vectors such that \( \ell_i - \ell_j = Z_i^T u_{ij} - Z_j^T u_{ji} \). The constant depends on the maximum 2-norm of these \( u_{ij} \) vectors.

**Proof:** At a high level, the proof below is divided into two main parts. First we need to show that the update rule—even though the underlying distribution is not Gaussian—serves as a good tool for tracking the true outcome distribution. After some algebraic manipulations, the problem reduces to a finding a high probability upper bound for norms
of weighted sums of noise vectors. To this end, we used the martingale version of the matrix Hoeffding inequality (Tropp 2012, Theorem 1.3).

Then, we need to show that the confidence ellipsoid shrinks fast enough that if we only choose actions whose cell intersect with the ellipsoid, we do not suffer a large regret. In the core of proving this, we arrive at a term where we need to upper bound \( \| \ell_i - \ell_j \|_{\Sigma_i} \), for some neighboring action pairs \((i, j)\), and we show that due to local observability and the speed at which the posterior covariance shrinks, this term can be upper bounded by roughly \( 1/\sqrt{t} \).

**Validity of the update**

We assume that \( p^* \), the true opponent strategy, is within some distance from our initial prior \( p_0 \), measured in \( \Sigma_0^{-1} \)-distance:

\[
\| p_0 - p^* \|_{\Sigma_0^{-1}} \leq 1.
\]

First we observe that the update can be rewritten in a cumulative form, to see how the parameters change from the initial prior \((p_0, \Sigma_0)\):

\[
\Sigma_t^{-1} = \Sigma_0^{-1} + \sum_{s=1}^t P_{Is}
\]

\[
\Sigma_t^{-1} p_t = \Sigma_0^{-1} p_0 + \sum_{s=1}^t P_{Is} X_s.
\]

Now let us investigate the \( \Sigma_t^{-1} \)-distance of \( p_t \) from \( p^* \!:

\[
\| p_t - p^* \|_{\Sigma_t^{-1}} = \left\| \Sigma_t \Sigma_0^{-1} p_0 + \Sigma_t \sum_{s=1}^t P_{Is} X_s - p^* \right\|_{\Sigma_t^{-1}}
\]

Now we decompose the samples \( X_s \) to mean and noise with the new notation \( X_s = p^* + \epsilon_s \), yielding

\[
\| p_t - p^* \|_{\Sigma_t^{-1}} = \left\| \Sigma_t \Sigma_0^{-1} p_0 + \Sigma_t \sum_{s=1}^t P_{Is} (p^* + \epsilon_s) - p^* \right\|_{\Sigma_t^{-1}}
\]

\[
= \left\| \Sigma_t \Sigma_0^{-1} p_0 + \Sigma_t \left( \sum_{s=1}^t P_{Is} \right) \sigma^{-1}_t p^* + \Sigma_t \sum_{s=1}^t P_{Is} \epsilon_s \right\|_{\Sigma_t^{-1}}
\]

\[
\leq \left\| \Sigma_t \Sigma_0^{-1} (p_0 - p^*) \right\|_{\Sigma_t^{-1}} + \left\| \Sigma_t \sum_{s=1}^t P_{Is} \epsilon_s \right\|_{\Sigma_t^{-1}}.
\]
We deal with the two resulting terms separately.

\[
\left\| \sum_t \Sigma_t^{-1} (p_0 - p^*) \right\|_{\Sigma_t^{-1}}^2 = (p_0 - p^*)^\top \underbrace{\Sigma_0 \sum_t \Sigma_t^{-1}}_{(I - \Sigma_0^{-1} \sum_s P_s)^{-1}} (p_0 - p^*) \leq \|p_0 - p^*\|_{\Sigma_0^{-1}} \leq 1.
\]

The second term is harder. Basically this is the term where we “pay the price” for not having started with a Gaussian distribution. We need to show that

\[
\left\| \sum_t \sum_{s=1}^t P_s \epsilon_s \right\|_{\Sigma_t^{-1}} = \left\| \sum_{s=1}^t \sqrt{\sum_t P_s} \epsilon_s \right\|
\]

is bounded with high probability. For any given action sequence, the above expression is a sum of independent random matrices. Now we recite a concentration inequality we need:

**Theorem 6.6 (Matrix Hoeffding Theorem (Tropp 2012, Theorem 1.3)).**

Consider a finite sequence \( \{X_k\} \) of independent, random, self-adjoint matrices with dimension \( d \), and let \( \{A_k\} \) be a sequence of fixed self adjoint matrices. Assume that each random matrix satisfies

\[
\mathbb{E} X_k = 0 \quad \text{and} \quad X_k^2 \leq A_k^2 \quad \text{almost surely.}
\]

Then, for all \( t \geq 0 \),

\[
P \left( \left\| \sum_k X_k \right\|_2 \geq t \right) \leq d \exp \left( -t^2 / 8\sigma^2 \right) \quad \text{where} \quad \sigma^2 = \left\| \sum_k A_k^2 \right\|_2.
\]

The above theorem can be extended to rectangular matrices, using the “dilation trick”\(^1\): for rectangular matrices \( B_k \in \mathbb{R}^{d_1 \times d_2} \), we use the theorem with

\[
X_k = \begin{pmatrix} 0 & B_k \\ B_k^\top & 0 \end{pmatrix} \in \mathbb{R}^{d_1 + d_2}.
\]

In our case, \( X_s = \sqrt{\sum_t P_s} \epsilon_s \). Also note that here we need the martingale version of the inequality, which also holds, according to Section 7 of Tropp (2012). After algebraic manipulations, we arrive at

\[
P \left( \left\| \sum_s P_s \epsilon_s \right\|_{\Sigma_t^{-1}} \geq \sqrt{\frac{1}{2} N \log \frac{M + 1}{\delta}} \right) \leq \delta.
\]

\(^1\) See remark 3.11 in Tropp (2012).
Putting together the terms we get that
\[ \|p_t - p^*\|_{\Sigma_t^{-1}} \leq 1 + \sqrt{\frac{1}{2}N \log \frac{M+1}{\delta}} \]
with probability at least \(1 - \delta\).

**Regret**

Now we turn our attention to calculating the regret of the algorithm that chooses the action that is chosen the fewest times so far among the actions whose optimality cells intersect with the current confidence ellipsoid. To accommodate the error for the outcome distribution not being Gaussian, we use the ellipsoid defined as
\[
\left\{ p : \|p - p_t\|_{\Sigma_t^{-1}} \leq 1 + \sqrt{\frac{1}{2}N \log \frac{M+1}{\delta}} \right\}.
\]

The regret in a turn results from choosing a suboptimal action. Let us assume wlog that the optimal action is action 1, the true opponent strategy is \(p^*\), and the chosen action is action \(k\). Then, the instantaneous regret is
\[ r_t = (\ell_k - \ell_1)^\top p^*. \]

Now if we pick a point \(p\) in the intersection of the cell of action \(k\) and the confidence ellipse, we can connect \(p^*\) and \(p\) with a line segment. That segment goes through the cells of, say, \(1 = i_0, i_1, \ldots, i_d = k\). Then we can write
\[
(\ell_k - \ell_1)^\top p^* = \sum_{j=1}^{d} (\ell_{i_j} - \ell_{i_{j-1}})^\top p^*
= \sum_{j=1}^{d} (\ell_{i_j} - \ell_{i_{j-1}})^\top (p^j - p^*),
\]
where we denote by \(p^j\) the point where our line segment intersects the boundary of cells \(i_{j-1}\) and \(i_j\). The above equation is true because for every \(j\), \((\ell_{i_j} - \ell_{i_{j-1}})^\top p_j = 0\). Now we upper bound, for every \(j\), the term
\[
(\ell_{i_j} - \ell_{i_{j-1}})^\top (p^j - p^*) \leq \|\ell_{i_j} - \ell_{i_{j-1}}\|_{\Sigma_t} \|p^j - p^*\|_{\Sigma_t^{-1}},
\]
with the help of Hölder’s inequality. We know from the previous section that \( \|p^j - p^*\|_{\Sigma^{-1}} \) can be upper bounded with high probability. It remains to upper bound the first term.

With the help of the local observability condition, we have

\[
\ell_{ij} - \ell_{ij-1} = S_i^T v_{ij,i-1} - S_{ij-1}^T v_{i-1,j},
\]

for some \( v_{ij,i-1}, v_{i-1,j} \), and thus the problem reduces to upper bounding \( \|S_i^T v_{i,i'}\|_{\Sigma_i} \) for all \( 1 \leq i,i' \leq N \):

\[
\|S_i^T v_{i,i'}\|_{\Sigma_i}^2 = \left\| \sqrt{S_i \Sigma_i S_i^T} v_{i,i'} \right\|_{2}^2 \\
\leq \|S_i \Sigma_i S_i^T\|_2 \|v_{i,i'}\|_2^2 \\
\leq \|S_i \Sigma_i S_i^T\|_2 V_{\text{max}}^2,
\]

where \( V_{\text{max}} = \max_{1 \leq i,i' \leq N} \|v_{i,i'}\|_2 \).

Putting everything together we have that the instantaneous regret at time step \( t \) is

\[
r_t \leq 2V_{\text{max}}K_i \sqrt{\frac{C_1}{n_{\min}} \left( C + \sqrt{\frac{1}{2} N \log \frac{M+1}{\delta}} \right)}.
\]

Since our algorithm picks the action that is chosen the fewest number of times, it ensures that \( n_{\min} \geq t/N \). Summing up the instantaneous regret for every turn we get the desired result

\[
R_T \leq C_2 \sqrt{TN \log MT/\delta} \quad \text{w.p.} \geq 1 - \delta.
\]
Setting $\delta$ to $1/\sqrt{T}$, we get the desired result.

### 6.5 Experiments

We first run extensive evaluations of BPM on various synthetic datasets and compare the performance against CBP (Bartók, Zolghadr, et al. 2012) and FeedExp3 (Piccolboni and Schindelhauer 2001). The datasets used in the simulated experiments are identical to the ones used by Bartók, Zolghadr, et al. (2012) and thus allow us to benchmark against the current state of the art. We also provide results of BPM on a dataset that was collected by Singla and Krause (2013) from real interactions with many users on the Amazon Mechanical Turk (AMT) crowdsourcing platform. We present the details of the datasets used and the summarize our results and findings in this section.

#### 6.5.1 Implementation Details

In order to implement BPM, we made the following implementation choices:

1. To use BPM-Least (see Section 6.3.2), we need to recover the current feasible actions. We do so by sampling multiple (10000) times from concentric Gaussian ellipsoids centred at the current mean ($p_t$) and collect feasible actions based on which cells the samples lie in. We resort to sampling for ease of implementation because otherwise we deal with the problem of finding the intersection between an ellipsoid and a simplex in $M$-dimensional space.

2. To implement BPM-TS, we draw $p$ from the distribution $N(p_{t-1}, \Sigma_{t-1})$. We then project it back to the simplex to obtain a probability distribution on the outcome space.

Primarily due to sampling, both our algorithms are computationally more efficient than the existing approaches. In particular, BPM-TS is ideally suited for real world tasks as it is several orders of magnitude faster than existing algorithms during all our experiments. In each iteration, BPM-TS only needs to draw one sample from a multivariate gaussian and does not need any cell decompositions or expensive computations to obtain high dimensional intersections.
6.5 Experiments

6.5.2 Simulated dynamic pricing games

Dynamic pricing is a classic example of partial monitoring (see the introduction), and we compare the performance of the algorithms on the small but not locally observable game described in Bartók, Zolghadr, et al. (2012). The loss matrix and feedback tables for the dynamic pricing game are given by:

\[
L = \begin{pmatrix}
0 & 1 & \cdots & N-1 \\
c & 0 & \cdots & N-2 \\
\vdots & \ddots & \ddots & \vdots \\
c & \cdots & c & 0
\end{pmatrix}, \quad H = \begin{pmatrix}
y & y & \cdots & y \\
n & y & \cdots & y \\
\vdots & \ddots & \ddots & \vdots \\
n & \cdots & n & y
\end{pmatrix}.
\]

Recall that the game models a repeated interaction of a seller with buyers in a market. Each buyer can either buy the product at the price (signal “y”) or deny the offer (signal “n”). The corresponding loss to the seller is either a known constant \(c\) (representing opportunity cost) or the difference between offered price and the outcome of the customer’s latent valuation of the product (willingness-to-pay). A similar game models procurement processes as well. Note that this game does not satisfy local observability. While our theoretical results require this condition, in practice, if the opponent does not intentionally select harsh regions on the simplex, BPM remains applicable. Under this setting, expected individual regret is a reasonable measure of performance. That is, we measure the expected regret for fixed opponent strategies. We also consider the minimax expected regret, which measures worst-case performance (pointwise maximum) against multiple opponent strategies.

Benign opponent While the dynamic pricing game is not locally observable in general, certain opponent strategies are easier to compete with than others. Specifically, if the stochastic opponent chooses an outcome distribution that is away from the intersection of the cells that do not have local observability, the learning happens in “non-dangerous” or benign regions. We present results under this setting for simulated dynamic pricing with \(N = M = 5\). The results shown in Figures 6.2a and 6.2d illustrate the benefits of both variants of BPM over previous approaches. We achieve an order of magnitude reduction in the regret suffered w.r.t. both the minimax and the individual regret.
Figure 6.2: (a,b,d,e) Comparing BPM on the locally non-observable game ((a,d) benign opponent; (c,e) hard opponent). Hereby, (a,b) show the pointwise maximal regret over 15 scenarios, and (d,e) show the regret against a single opponent strategy. (c) shows the effect of a misspecified prior. (f) is the performance of the algorithms on the real dynamic pricing dataset.
6.5 EXPERIMENTS

hard opponent For the same problem, with opponent chooses close to the boundary of the cells of two non-locally observable actions, the problem becomes harder. Still, BPM dramatically outperforms the baselines and suffers very little regret as shown in Figures 6.2b and 6.2e.

Effect of the Prior We study the effects of a misspecified prior in Figure 6.2c. As long as the initial confidence interval specified by the prior covariance is large enough to contain the opponent’s distribution, an incorrectly specified prior mean does not have an adverse effect on the performance of BPM. As expected, if the prior confidence ellipse used by BPM does not contain the opponent’s outcome distribution, however, the regret grows linear in time. Further, if the prior is very informative (accurately specified prior mean and tight confidence ellipse), very little regret is suffered.

6.5.3 Results on real data

Dataset Description We simulate a procurement auction based on real data. Parameter estimation was done by posting a Human Intelligence Task (HIT) on the Amazon Mechanical Turk (AMT) platform. Motivated by an application in viral marketing, users were asked about the price they would accept for (hypothetically) letting us promote material to their friends on a social networking site. The survey also collected features like age, geographic region, number of friends in the social network, activity levels (year of joining, time spent per day etc.). Note that since the HIT was just a survey and the questions were about a hypothetical scenario, participants had no incentives to misreport their responses. Complete responses were collected from approx. 800 participants. See Singla and Krause (2013) for more details.

The Procurement Auction We simulate a procurement auction by playing back these responses offline. The game is very similar in structure to dynamic pricing, with the optimal action being the best fixed price that maximized the marketer’s value or equivalently, minimized the loss. We sampled iid from the survey data and perturbed the samples slightly to simulate a stream of 300000 potential users. At each iteration, we simulate a user with a private valuation generated as a function of her attributes. We discretized the offer prices and the private valuations to be one of 11 values and set the opportunity cost.
of losing a user due to low pricing to be 0.5. Thus we recover a partial monitoring game with 11 actions and 11 outcomes with a 0/1 feedback matrix.

**RESULTS** We present the results of our evaluation on this dataset in Figure 6.2f. Notice that although the game is not locally observable, the outcome distribution does not seem to be in a difficult region of the cell decomposition as the adaptive algorithms (CBP and both versions of BPM) perform well. We note that the total regret suffered by BPM-LEAST is a factor of 10 lower than the regret achieved by CBP on this dataset. The plots are averaged over 30 runs of the competing algorithms on the stream. To the best of our knowledge, this is the first time partial monitoring has been evaluated on a real world problem of this size.

### 6.6 SUMMARY

In this chapter, we introduced a new family of algorithms for locally observable partial-monitoring problems against stochastic opponents. We enriched the model of partial monitoring by incorporating prior information about the outcome distribution in the form of a confidence ellipsoid. We explicitly track the true outcome distribution, instead of tracking loss differences. This is a novel insight compared to existing algorithms. This approach not only eases computational overhead but also helps achieve low regret by being able to transfer information between actions. In particular, BPM-TS runs orders of magnitude faster than any existing algorithms, opening the path for the model of partial monitoring to be applied on realistic settings involving large numbers of actions and outcomes. Our experiments substantiate our theoretical and computational performance claims.
DISCUSSION AND CONCLUSION

While there is an explosion of streaming data generation in the form of text, pictures and videos, reliable labeled data remains sparse. Online learning with partial feedback continues to be an interesting and active area of research with the need for principled and scalable algorithms growing by the day. The ability to actively influence the labelled data collection policy has made these algorithms ubiquitous in many web based and online decision making domains.

Existing work in this field has either been heuristically motivated or have been less effective in practice than in the theoretical results. We believe the techniques and approaches presented in this thesis offer the best of both worlds in this regard.

Some of our common themes through all the methods discussed in this thesis are:

1. **Using uncertainty**: We believe that point estimates do not convey the whole story. We pursue a Bayesian approach in all our algorithms and this allows us to reason about the uncertainty in our predictions. We actively seek to reduce the uncertainty around potentially valuable candidate actions. Our selection rules for both CGPRANK and GP-Select use the uncertainty weighted against the means in order to decide on the best action to pick in each round. BPM uses the uncertainty inherent in the confidence ellipsoid to track the true outcome distribution.

2. **Modeling similarity**: We have shown over the course of this thesis that effectively modeling similarities among actions and/or users provides a boost over other techniques. In fact, we even showed that a naive and coarse grained similarity model (clustering) provided a significant lift over algorithms that do not take such similarity into account. Our algorithms CGPRANK and GP-Select model such similarities using a kernel function and model function smoothness using Gaussian Processes. BPM is also able to effective share feedback via its Bayesian updates.
3. **Using the right metric:** As we argued in Chapter 3, RMSE is not the right metric to measure the performance of recommendation algorithms. We use regret to measure the performance of our algorithms and define appropriate optimal benchmarks (best fixed action, best fixed policy, Optimal hindsight policy).

4. **Theoretical guarantees:** We provide strong performance guarantees of the algorithms we presented in the form of regret bounds. In all the cases, it matches best existing known results. While we analyze the cases as if the utility function $f$ was drawn from a benign distribution (drawn from a GP or Gaussian prior), the bounds hold for $f$ drawn a much richer and more expressive class of functions.

5. **Computational efficiency:** We strived to make our algorithms scalable. For CGPRANK and GP-Select, the lazy variance updates improved performance significantly. Specifically for GP-Select, we showed almost 40X speedup using the failsafe lazy variance updates.

7.1 **SUMMARY**

Our goal during the course of the thesis was to design theoretically sound and computationally efficient learning algorithms to learn under the presence of partial or/and proxy interaction feedback. We summarize our contributions here:

7.1.1 **Statistically efficient web scale recommendations**

In Chapter 3, we addressed the problem personalized list recommendation on the web. We formulated the problem of personalized recommendations as a contextual multi-arm bandit problem where we receive a context $z_t$ in each round $t$ and have to pick an ordered list $\mathcal{L}_t$ in response. We highlighted some insights drawn from interactions of millions of users on thousands of items on a real world recommender system. Since there is a need to generalize from sparse and partial feedback, we devised mechanisms to share feedback across (1) positions, (2) items, and (3) users. We presented CGPRANK which builds on existing Gaussian process bandit algorithms and is a statistically and computa-
tionally efficient recommendation algorithm. We proved efficient regret bounds that under surprisingly showed that under natural assumptions of position dependent clickthrough rates, the average per slot regret reduces with increased list size. We demonstrated the effectiveness of CGPRank on large scale recommender systems based on (1) Google books dataset and (2) Yahoo! Today module news recommendation dataset.

7.1.2 AVID — AdaptiveValuableItemDiscovery

In Chapters 4 and 5, we introduced and formulated the problem of AVID. In its most general setting, AVID generalizes the 0/1 knapsack problem to the setting where values of items are initially unknown and are revealed only on selection of the item. AVID also generalizes many online learning problems with partial feedback including the classical multi–arm bandit and budget bandit settings. We discussed the setting where in addition to value, the objective function also had a preference for diversity in the selected subset. We presented GP-Select and provided performance guarantees for its performance on budgeted subset selection problems. The experiments showed the efficacy of GP-Select on two real world tasks – airline price update and drug design. We also discussed methods to speed up GP-Select and demonstrated the improvements on the Yahoo! Today module news recommendation dataset with almost 40X speedup.

7.1.3 Bayesian partial monitoring

In Chapter 6, we discussed the problem of stochastic partial monitoring – a general framework for online learning with partial feedback. Under mild assumptions on the availability of a prior distribution of outcomes, we develop BPM– a family of new efficient algorithms. The main idea in BPM is the tracking of the outcome distribution with an ellipsoid around the estimated outcome distribution. We present two algorithms – BPM-Least and BPM-TS in the BPM family and show near optimal regret rates for BPM-Least. We document the improvements provided by the two algorithms over existing approaches on both synthetic and real world data.
7.2 FUTURE DIRECTIONS

The work in this thesis also raises interesting further questions and opens up avenues for future research problems. We detail a few of them below.

7.2.1 Contextual recommendations

1. CGPRANK is designed to work for contextual list recommendation tasks. In proving the bounds for the regret, we crucially assumed that the items in the list do not interact with each other. This independence assumption let us factorize the feedback of an item into its individual relevance term and the effect due to its position. Relaxing this assumption would be an useful and interesting extension to CGPRANK. In our experiments, we did test the performance of CGPRANK when this independence assumption was relaxed. CGPRANK continued to perform better than baselines. But it would be interesting to analyze the regret of an efficient recommendation algorithm that systematically accounts for these intra–item interactions.

2. When we have access to features, we assume that the features for the contexts and the items are fixed throughout the time horizon. However, if we were to assume a prior on the features and perform full Bayesian inference on the features as well, the resulting algorithm will be better suited to handle user drift and popularity of items over time.

7.2.2 Adaptive Valuable Item Discovery

1. What happens in the AVID setting when even the item costs are unknown a priori? This is a practical problem where we only have beliefs about the testing costs and do not know the exact costs until the tests are actually carried out. One approach might be to model the costs of items as a function drawn from a Gaussian process and through careful selection of items, learn both the value and the cost function.
2. Adding a contextual component to AVID and the GP-Select algorithm will be a straightforward extension given the research in this thesis. It would be useful in personalized collection building tasks like online movie/music libraries and playlists.

3. AVID with diversity preference in the objective can be extended to handle any submodular function (or combinations of submodular functions). As long as the objective is a linear combination of a unknown modular function and a known submodular term, the analysis of GP-Select remains valid. One can then envision applications with complex submodular function modeling matroid constraints or $p$-independence systems.

7.2.3 BPM

1. An immediate open problem in the analysis of algorithms in the BPM problem is the regret guarantee for the Thompson sampling version of our algorithm, BPM-TS.

2. Another immediate extension would be that of incorporating side information in the partial monitoring game and analyzing the regret of algorithms in the BPM family.

3. A well studied problem in online learning with limited feedback is that of Dueling bandits which can be set up as a special case of partial monitoring. It would be interesting to study the regret of this game using BPM and compare it against standard results (Yue, Broder, et al. 2012; Ailon et al. 2014).

I believe the problems described here offer immediately useful and applicable research directions while also posing sufficient challenges for the researcher. I am personally passionate about all these problems and hope to continue working on these and other related interesting directions in the field of online learning with partial feedback.


Bibliography


Bibliography


Paquet, Ulrich and Noam Koenigstein (2013). “One-class collaborative filtering with random graphs.” In: *Proceedings of the 22nd Interna-

---

119


Bibliography


COLOPHON

This document was typeset in \LaTeX{} using the typographical look-and-feel classicthesis. Most of the graphics in this thesis are generated using pgfplots and pgf/tikz. The bibliography is typeset using biblatex.